パソコンプログラマのためのクリエイティブマガジン

平成3年1月2日第三種郵便物認可　1992年8月1日発行(毎月1回18日)　第2巻第8号通巻20号

月刊プログラマーズ・ページ

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1992 8 AUGUST

定価980円



特集 進化するWindowsの開発環境

誰でもできるWindowsプログラミング

C++化したWindowsの処理系

付録:5"2HDフロッピーディスク

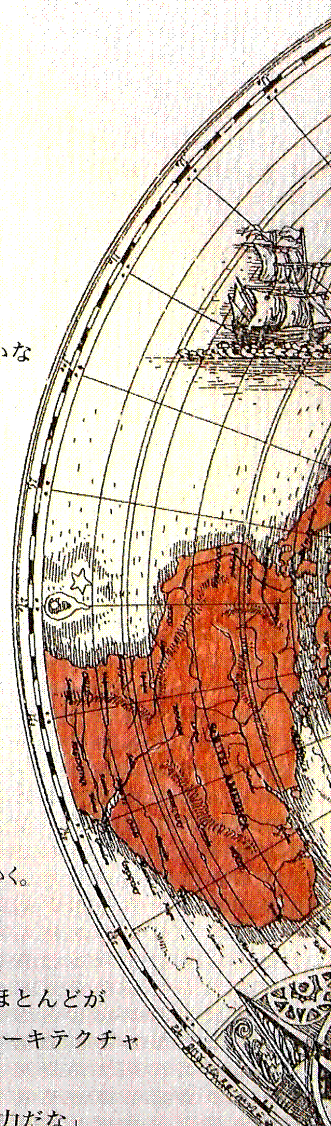
本誌掲載プログラム／綴じ込み付録テキストファイル「Microsoft C/C++ Ver.7.0ファウンデーションクラスライブラリ一覧」／クイズ・ランダムスクリーン解読ツール／mmm系WTERM用オートパイロットマクロ「wtmmn」／mmmオフラインシミュレータ「ハイパーmmm」／斜め読み支援ツール「Askance」／EMS常駐リモート操作システム「パックマン」(試用版)

SE
Shoeisha Co., Ltd.



世界標準物語　第三話

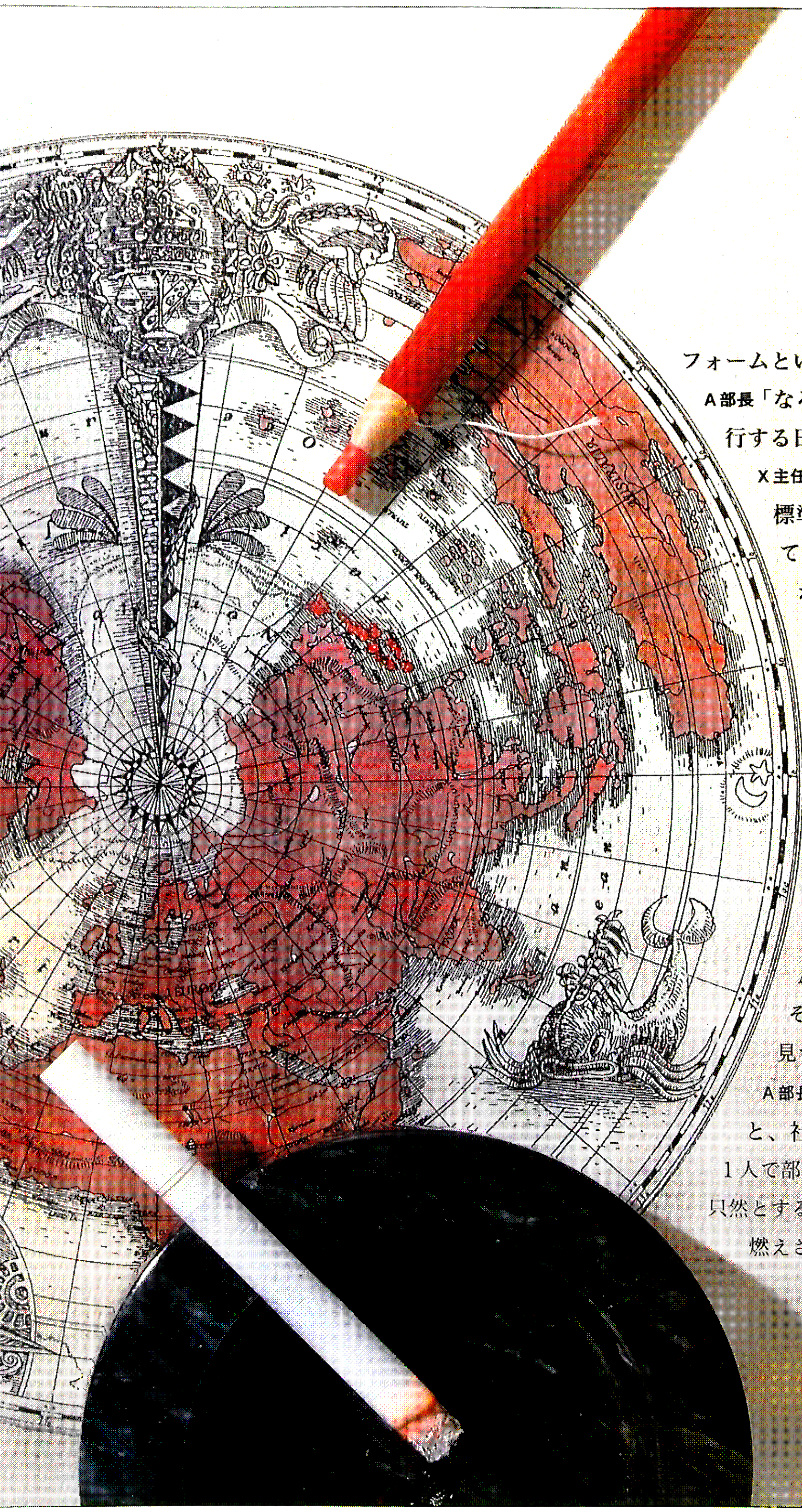
14本目の煙草



とある企業のオフィス。二人の男がテーブルをはさみ、真剣な顔で向き合っている。
テーブルの上の灰皿には、すでに12本の煙草の吸殻がある。
A部長「それじゃあキミはどうしても、AXがいいというんだね？」
X主任「はい、どうしてもAXがいいんです。今回だけは一歩も譲れません」
短い沈黙。A部長、13本目の煙草を取り出し、おもむろに火をつける。
A部長「しかし、うちの社内でAXを使っているのは、まだキミを含めて数人しかいないんだ。他のものが使いこなせるかね」
X主任「使いこなせるもなにも、操作自体は全く変わらないんです。違うのは、パソコン内部のアーキテクチャです。AXは、事実上の世界標準機であるIBMPC/ATと同じオープンアーキテクチャを採用しているので、世界に6万本以上といわれるIBMPC用ソフトがそのまま使えるんです。今、うちの社内で使っているパソコンは、日本では一応、標準機ということになってますが、閉ざされた、メーカー独自のアーキテクチャを採用しているので、そういうことはないんですよ」
A部長「IBMPCは、そんなに世界中で使われているのか？」
X主任、待ってましたとばかりに、一枚の世界地図をとりだす。
X主任「見てください。アメリカはもちろん、カナダ、メキシコ、ブラジル、イギリス、フランス、ドイツ、ソ連、中国…………。IBMPCとその互換機は日本以外のほとんどの国でトップシェアを占めているのが現実です」
X主任、つぎつぎに国名をあげながら、猛スピードでその国の位置を赤鉛筆で塗りつぶしていく。
A部長、思わず大きな溜息とともに煙を吐く。
A部長「うーん、これはすごい。日本はまさに世界の異端児ってわけだ」
X主任「MS-Windows、LAN、DTPソフトなど、現在我々が使っているソフトのほとんどが海外でIBMPC用として開発されたものです。日本語への移植も、AXなら基本アーキテクチャが同じなので即座に行えるというわけです」
A部長「GUI環境を実現するWindows3.0対応ソフトがすぐ使えるのは、確かに魅力だな」
X主任「それにAXはVGAに対応していますからね。これらの画像は、640X480ドットの高品質な動画やアウトラインフォントを扱えるVGAモードでなくちゃ、世界のマルチメディアの流れにはついていけませんよ」
A部長、煙草を揉み消し、身を乗り出す。
X主任「オープンアーキテクチャのAXこそが、ネットワークコンピューティングを可能にする共通プラット

世界の標準機を、日本の標準機に。
我々が、パソコンの新しい時代を拓きます。

Acer　Canon
MITSUBISHI　NCR　OKI　olivetti　SANYO



THE ARCHITECTURE EXTENDED－AX

フォームといえるんです」

A部長「なるほど、日本市場が海外市場と同時進行する日も近いわけだ」

X主任「部長、事実上の世界標準機が、日本標準機になる日が、もうすぐそこまで来ているんです。新しい時代に備えて、今が決断の時です」

再び沈黙。A部長、14本目の煙草に火をつける。

その時ドアをノックする音がして、部長の秘書が現われる。

秘書「部長、会議のお時間です。もう社長がいらしてますので急いでください」

A部長「しまった、もうそんな時間か。すぐに行くと伝えてくれ」

A部長、慌てて立ち上がりドアに向かう。そこで振返って、X主任の目をまっすぐ見つめて言う。

A部長「よしわかった。……AXしかないと、社長に言うことにしよう」

1人で部屋に残されたX主任は信じられぬ思いで呆然とする。が、すぐに満面に笑みを浮かべると、燃えさしの14本目の煙草を灰皿に押しつけた。

※MS-Windows 3.0は米国Microsoft社の登録商標です。
※IBM PC、IBM PC/AT、VGAは、米国IBM社の商標です。

CASIO HEWLETT PACKARD HITACHI KYOCERA

SHARP SONY UNISYS

社名アルファベット順

お問合わせ：株式会社 翔泳社
TEL. 03-3263-0447

プログラム開発 LSI C-80 (MS-DOS) からデバッグ LDB-80X まで

LDB-80X

LDB-80Xは、LSI C-80(MS-DOS)で開発したソフトウェアのデバッグ環境を、MS-DOS上で実現するためのソースレベル・クロスデバッガです。これにより、プログラム環境からデバッグ工程まで、アセンブラを意識することなくC言語レベルでのプログラム開発が行えますので、大幅にソフトウェアの生産性が向上します。

■LDB-80Xには、動作モードとして以下の3種類のモードで動作します。
1. 64180CPUボードを使う180モード
2. V30CPUの8080エミュレーションを使うV30モード
3. ソフトウェア・エミュレーションモード

■Turbo-Vシステム、またはTurbo-180IIシステムと併用すれば、CP/M-80のアプリケーションソフト(.COMファイル)のデバッグが可能です。

■LDB-80Xは、操作性に優れたウィンドウ指向デバッガです。5つのウィンドウ(コード、レジスタ、ウォッチ、コマンド、及びオプションウィンドウ)とポップアップ・メニューがあります。

■実行中のユーザ・プログラムを任意の時点で強制的にブレークさせ、制御をLDB-80Xに戻すことができます。

■複数のコマンドを組み合わせて、新しいコマンドを構築するマクロコマンド機能があります。

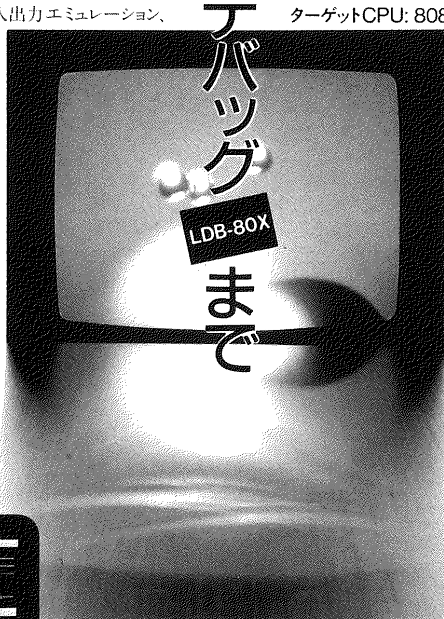
■LDB-80XからMS-DOSのコマンドを起動する、サブプロセス機能を持っています。

■その他、LDB-80Xには、コマンドウィンドウの表示コントロール(カーソル制御、画面消去等)、I/Oの入出力エミュレーション、メモ、オプション設定、ロギング、バッチ等のコマンドが用意されています。

[LDB-80Xの走行環境]
- MS-DOS Ver.2.11以上、384KByte以上必要
- NEC PC-9800シリーズ(PC-98XA, XL, XL², RLのハイレゾリューションモードも含む)
- EPSON PC-286シリーズ/PC-386シリーズ

LSI C-80(MS-DOS)+LDB-80X

セット価格128,000円



LSI Cコンパイラ

- 高速、コンパクトなコードを生成
- レジスタの最適自動割り付け
- レジスタによるパラメタ渡し
- アセンブリ言語の埋め込み記述可
- ANSI準拠・プロトタイプ宣言を完全サポート
- ROM化に最適

LSI C-80

ターゲットCPU: 8080/Z80/64180

■MS-DOS版クロス ……………………価格98,000円
- アセンブラ、リンカ付き
- UNIXのccと同様の使いやすいコンパイラ・ドライバ
- UNIXコンパチ・プログラム保守ツールmake
- ライブラリ、コンパイラドライバ、プリプロセッサのソース付き
- MS-DOS Ver.2.11以上、384KByte以上必要

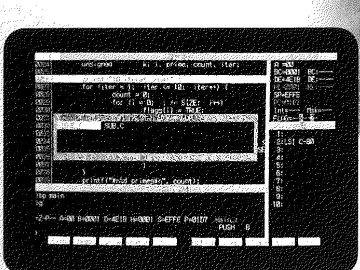
■UNIX版クロス ……………………価格398,000円より
- NEWS、Sun-3、vsxシリーズ、Kurama、SX-9100シリーズ、LUNA、HP9000モデル300シリーズ、Σstation200、Micro VAX、VAX-11　※その他の機種については、ご相談に応じます。

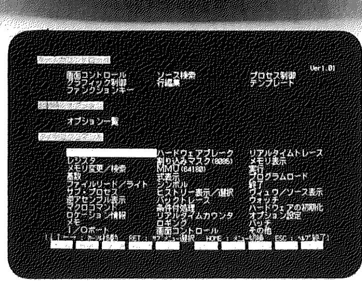
■CP/M版セルフ ……………………価格98,000円
- CP/M-80 Ver.2.2以上、メモリ60K CP/M以上必要

LSI C-86

ターゲットCPU: 8086 ……………………価格48,000円
- アセンブラ、リンカ付き
- プログラムの実行をモニタするツール
- ANSI標準規格で定義された関数のうち多バイト文字関係と<locale.h>を除く、すべての関数を用意、ライブラリーのソース付き
- ROM化のための初期設定モジュールのソース付き
- MS-DOS Ver.2.11以上、384KByte以上必要

※表示価格に消費税は含みません。

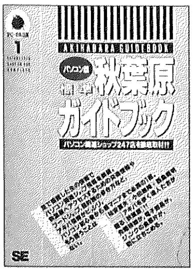




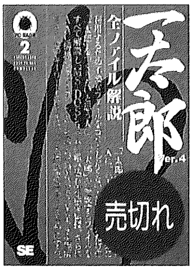
■C言語プログラミング入門・活用セミナー
5日間コース毎月定期開催
- セミナー日程(予定)
  毎月第2または第3週の月曜日から金曜日の5日間(9:30〜17:00)
- 開催場所
  東京
- 受講料
  70,000円(税別)
  「LSIC-86」、昼食含む

エル・エス・アイジャパン株式会社
〒151 東京都渋谷区千駄ヶ谷1丁目8番14号
☎(03)3404—1319・FAX(03)3478—0576

〈資料請求番号002〉



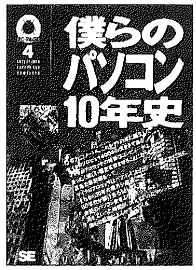
1 パソコン版標準秋葉原ガイドブック
定価1200円



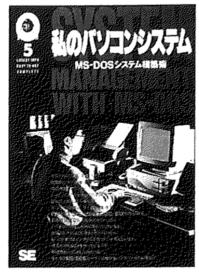
2 一太郎Ver.4全ファイル解説
定価1200円



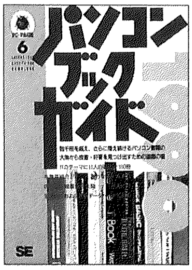
3 林檎百科
定価1200円



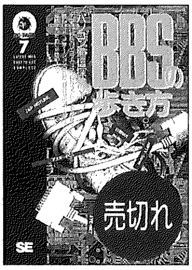
4 僕らのパソコン10年史
定価1200円



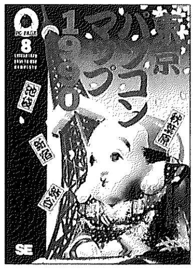
5 私のパソコンシステム
定価1200円



6 パソコンブックガイド1990
定価1200円



7 BBSの歩き方
定価1200円



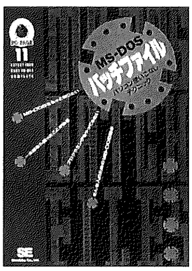
8 東京パソコンマップ1990
定価1200円



9 MS-DOS3.3全ファイル解説
定価1200円



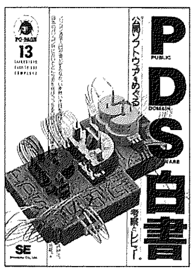
10 パソコンを思想する
定価1200円



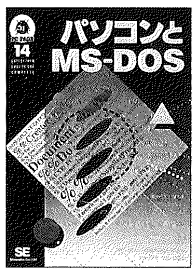
11 MS-DOSバッチファイル
定価1500円



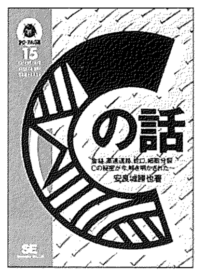
12 プロフェッショナルのパソコン利用術
定価1200円



13 PDS白書
定価1500円



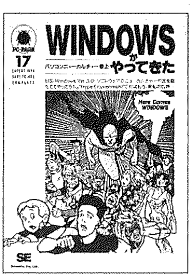
14 パソコンとMS-DOS
定価1200円



15 Cの話
定価1500円



16 パソコン見栄講座
定価1500円



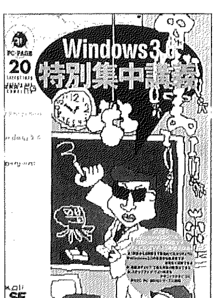
17 WINDOWSがやってきた
定価1200円



18 Macintoshグラフィック工房
定価1500円



19 日本語入力読本
定価1500円



20 Windows3.0特別集中講義
定価1500円



21 フリーソフトウェア派宣言
定価1500円

最新刊

NOW PRINTING

26
Mac派
DTP倶楽部
マックでミニメディアを作るための実践的解説書。

●定価1,600円・SE編集部編

※定価はすべて税込価格となっております。

翔泳社の本は全国どの書店でもお求めになれます。店頭にない場合は「至急」とお申し出の上ご注文いただければ10日位でお届けできます。

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

特集
進化するWindowsの開発環境

表紙イラスト：日暮修一
表紙 AD：佐々木幹夫(SE)

広告索引　資料請求番号

人材募集広告

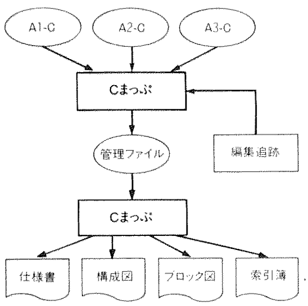
（掲載順）

1992 AUGUST No.20

今、C言語をお使いのあなたへ！

あなたは他人の作ったソースファイル中に何が定義してあるのか、またどの様な構成になっているのかすぐ理解することができますか？当社製作の〝Cまっぷ〟を使用すればすぐに理解することができます。

Aシステム



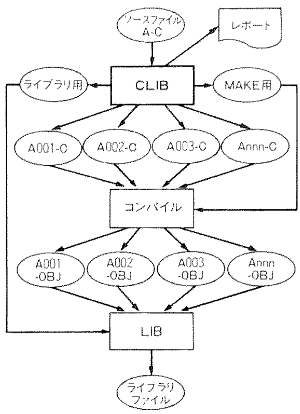
Cまっぷ ポイント

①直接ソースファイルを見ながら流れを確認→他の関数、ラベル変数へのタグジャンプも使い馴れたエディタソフトにて使用可能。
②ソースファイルを編集させずに調査したい関数だけに関するプログラム内容をリスト出力、CRT表示、ファイル出力可能。
③ソースファイル内の関数一覧、定義関数による構成図、ブロックチャート図、メンテナス用資料、関数索引簿等のドキュメント資料を連続用紙、手差し用紙、ファイルに出力可能。
④MAKE用ファイル、プロトタイプ宣言ファイルの作成やライブラリ作成用にソースファイルの分割処理も可能。
⑤関数単位の変更状況やオーバーレイー構造化の為の重複調査資料も作成可能。
⑥メンテナンス作業を行った場合のメモ情報も付加でき、管理する事が可能。

※この他にもプログラム解析用の画面、及び資料作成について色々なメニューを用意しており、マニュアルの強化も行ないました。

（ サーチ専用ソフト（サーチ）…………￥20,000（税別）
全てを含んだ（Cマップ）……………￥50,000（税別） ）

C言語用定義関数別のライブラリー化作業支援ソフト"CLIB"



CLIB ポイント

1本のソースファイル中にあるいくつかの関数を個別に分割しライブラリ化しておけば必要最小限の関数によって実行ファイルが作成できます。しかし、その分割作業をエディタを使用し行なうには大変な時間と労力を必要とします。そこで、本ソフトを使用すれば、それらを自動的に行なわせる事ができ、しかもグローバル変数も2本目以降は、外部宣言に置き換えてくれます。また、分割したもののコンパイル用MAKEファイル、及び関数の引数等だけを抽出したレポート（関数用コメント含む）出力ライブラリ化用パラメータファイルの出力指定も可能となっています。このように、あなたが完成した関数のライブラリ化を支援致します。

CLIB 機能

①HELP機能　②コマンド入力により実行可能

（ ライブラリー化支援ソフト（CLIB）…￥9,800（税別） ）

PDS Plan Do See

開発・販売元
㈱プランドシー

〒521-12 滋賀県神崎郡能登川町佐生19-1
TEL（0748）42-6528
FAX（0748）42-6458

〈資料請求番号003〉

本格派Cプログラマを目指す、すべての初心者へ‼



MS-C入門

●定価2,600円　A5判・380頁
吉川敏則　著

本書は、初心者を対象にしたものですが、
初心者が初級のままとどまるのではなく、
中級のレベルに到達することを目標としています。
これまでC言語の入り口で挫折していた人も、
詳細に書かれたこの一冊を手に入れることにより、
Cプログラマへの道が開かれます。

SE Shoeisha Co., Ltd. 株式会社翔泳社 〒102東京都千代田区平河町2-4-14平河町KSビル TEL.03-3263-0447（代表）
翔泳社の本は全国どの書店でもお求めになれます。店頭にない場合は「至急」とお申し出の上ご注文いただければ10日位でお届けできます。

WINDOWS Programming

特集

進化するWindowsの開発環境

1 誰でもできるWindowsプログラミング

2 C++化したWindowsの処理系

WINDOWS Programming

Windows 3.0が発売され，世間の注目を集めてから早1年以上が過ぎた．しかし，いまだに国産のWindows対応アプリケーションは数えるほどしか出荷されていない．これらの理由として，国産マシンの貧弱なグラフィックス環境，Windowsプログラミングの学習が困難であることに加えて，Windowsに対応した優秀なプログラミングツールが不足していたことがあげられる．しかし，日本語スタートアップガイド付属のVisualBasic，PC-9801版Borland C++ & Application Frameworks 3.0などの処理系が発売され，Windowsプログラミング環境は大幅に改善されてきている．この特集では，これらの進化した開発環境の実力を探ってみたい．合わせて，最新のWindowsプログラム開発環境であるMS-C/C++ 7.0の紹介も行う．



特集

進化するWindowsの開発環境

1.►►►►►►► 誰でもできるWindowsプログラミング

もっとも容易にWindowsプログラムを開発できるツールとしてアメリカで絶賛されているVisualBasicが，ようやく国内でも発売された．日本語スタートアップガイド付属という形であり，完全な日本語化はされていないものの，日本語アプリケーションの作成にはとくに問題はない．このVisualBasicに最近Microsoft社から発売されたProfessional Tool Kitを加えて，本格的なWindowsアプリケーションの開発方法を指南する．実行スピードが遅いといわれるVisualBasicでも，やり方次第で十分実用になるアプリケーションが開発できることがわかるだろう．

スクリーンセーバーとカスタムコントロールの制作

Visual Basicによる実践的プログラム開発

川俣　晶

VisualBasicを用いた実践的プログラム開発として，Professional Tool Kitに付属するControl Development Kit（CDK）を用いてスクリーンセーバーを作成した．CDKを利用することで，本格的なアプリケーション開発が可能になることを知っていただきたい．

概　要

本稿では，Visual Basicによって，実用的なプログラムを開発する手順とノウハウを解説する．それにあたって，2部構成とした．それぞれ，想定する読者層と，内容は以下のとおりである．なお，本稿では，Visual BasicをVBと略する．

第1部「VBによるアプリケーションの開発」

VBの基礎知識を持っている人を対象としている．VBで実用的なアプリケーションを開発するステップを説明する．日曜プログラマの方にも，ぜひ読んでいただきたい．

第2部「CDKの活用」

これは，CとSDKによるプログラム開発と，VBの知識を持っている人を対象とする．プロフェッショナルなプログラマか，腕に覚えのあるプログラマの方々だけを対象としている．VBを使って思い通りのプログラムを開発する方法を述べる．

Visual Basicについて

Windows 3.0が出始めた頃には，Windowsのプログラミングは非常に難しく複雑であるといわれていたものである．しかし，現在，「Windowsのプログラミングなど難し過ぎてできるわけがない」と主張する人は少なくなった．

この背景には，プログラミングのための環境の劇的な改善がある．当初，MS-CとSDKの組み合わせでしか開発できなかったものが，C++とクラスライブラリの利用はあたりまえとなった．また，Quick C for Windowsのような扱いやすい統合された環境も登場した．実用性や安心感の点はともかく，マイクロソフト以外のソフトハウスから言語やツールも入手可能となった．

さらに，見逃すことができないのが，ここで解説するVisual Basicの登場と普及である．Visual Basicは機能的にはC/C++とSDKに比べれば，かなり速度も遅く，自由度も低い．しかし，その簡便性によって，Windows環境でのプログラミングの敷居を一気に低くしたのである．

しかも，ただ敷居を低くしただけではない．グラフィカルで見栄えのよいプログラムをMS-DOSベースでプログラミングするよりも，はるかに少ない手間で作成（もはや開発という言葉すら似合わない）できるのである．

もはや，Visual Basicを使えば，Windowsアプリケーションを作ることは，難しいことではない．まさに，日曜プログラマの復権という言葉があてはまるであろう．

さて，ここまでは，参考文献1で述べられている．問題は，その先である．趣味のプログラミングではなく，ハイレベルなアマチュアや，プロのために，VBは役に立つのだろうか．これが，本稿のテーマである．

今回は，意地悪く，VBにとって不利なテーマを選んでみた．Windowsの

スクリーンセーバーである．VBにとって，システムの深部にタッチするプログラムは不得意である．WindowsのAPIやDLLを自由に呼び出せるとはいえ，いろいろな制約から，すべての機能が自由に使えるわけではない．そして，スクリーンセーバーのための，「キーボードまたはマウスから一定時間入力がないとき」を判定するルーチンを組むには，まさにVBでは扱えないシステムの深部にタッチするしかない．これがクリアできれば，VBの実用性は疑う余地もなくなるはずである．

結論をいえば，意外というか，嬉しい誤算というか，あっけなくクリアできてしまった．

要は，CDKの活用である．CDKとは，Control Development Kitの略で，VBのコントロールをC＋SDKで開発するためのキットである．これを使えば，Cやアセンブラで書いたルーチンを，自由に組み込むことができ，VB上から簡単に扱える．ここでミソなのは，単に組み込めるというだけでなく，簡単に扱えるということである．CDKによって，汎用的なカスタムコントロールを作っておけば，大多数の普通のVBプログラマが，標準装備の機能と同等に，それを利用してプログラムの開発を行うことができるのである．

ここで，発想の転換をしてほしい．最初から，VBで凝ったプログラムを記述することは不可能である．しかし，プログラムのユーザーインターフェースの部分は，ほとんどVBで記述可能である．ならば，メインルーチンをCDKとC/C＋＋とSDKで記述し，ユーザーインターフェースの部分だけVBで記述したらどうであろうか．機能的にはC/C＋＋とSDKで組んだものに匹敵するものが，はるかに少ない時間で開発できるはずである．

本稿の第1部は，日曜プログラマ的な視点から，すでにCDKによって作られたカスタムコントロールを利用する立場で解説する．

第2部は，CDKも自分で利用する立場から，ハイレベルの読者に向けて解説する．

第1部 VBによるアプリケーションの開発

今回は，VBにはできそうもないテーマとして，スクリーンセーバーを取り上げた．そのために，CDKによって，カスタムコントロールを1つ作成した．VSAVER1.VBXというファイルがこれである．このファイルが1つあれば，あとは，自分の好きなスクリーンセーバーをVBで作成することができる．しかし，これを使って，単なるスクリーンセーバーを作るだけではおもしろくない．同時に，VBで，複数のモジュールがインストールできるスクリーンセーバーのシステムまで組んでしまった．ここまでくると，敵はアフターダークである．みなさんが，これを読んで，おもしろいスクリーンセーバーをたくさん作ってくれれば，スクリーンセーバーのデファクトスタンダードになるかもしれないと考えている（半分は冗談だが，半分は本気である）．

スクリーンセーバーコントロールの仕様

さて，利用する前に，カスタムコントロールの仕様を見てみよう．

まず，このカスタムコントロールは，CDKのcircle3というサンプルソースをベースにしているので，このcircle3の持っている機能はすべて持っている．しかし，今回のプログラムにはあまり関係ないので，スクリーンセーバー用の機能だけ解説しよう．

まず，ファイル名は，vsaver1.vbxである．デフォルトのコントロール名はScreenSaverである．

このコントロールは，タイマのように「不可視」で使うことを想定している．

以下のようなプロパティとイベントを持っている．

プロパティ:InSaving

データ型はENUMである．0か1の値を代入できる．これに，1を入れておくと，StartSaverのイベントが発生しなくなる．つまり，スクリーンセーバーが起動中に，再度起動されないためのスイッチとして使うことができる．スクリーンセーバーがスタートするときに，これを1にして，終了したときには0を入れる．

プロパティ:WaitSeconds

データ型はSHORTである．キーまたはマウスの入力がなくなってから，何秒後にStartSaverイベントが発生するかを設定する．60と設定すれば60秒である．1より小さい値は無効である．

プロパティ:SampleRate

イベントを発生させるべきかどうか，調査するタイミングを指定する．スクリーンセーバーを起動させるべきタイミングで，Windowsカーネルが何かのイベントを発生してくれるわけではないので，タイマで一定時間毎に，セーバースタートの条件に達したかどうかをチェックしなければならない．そ

Windows プログラミング用語集
DPMI（DOS protected mode interface）　DOSエクステンダなど複数のアプリケーションからプロテクトメモリを利用するためのインターフェース．

のインターバルを秒単位で指定する．この値が小さければ敏感だが，システム全体のCPUパワーを食う．大きくすれば鈍感になるが，CPUパワーは食わなくなる．デフォルトは1秒だが，とくに変える必要はないだろう．

イベント:StartSaver

次のどちらかの条件が満たされたとき，発生するイベント．

(1)キー入力もマウス入力もなく，WaitSecondsプロパティで指定した時間が経過したとき

(2)マウスポインタを画面右上隅に置いたとき

引数はとくにない．

また，これによって，何かが起こるわけではなく，スクリーンセーバーの動作を行うルーチンを起動するのは，プログラマの責任である．また，このイベントによって，InSavingプロパティが自動的に書き換えられるわけではない．実際のセーブ動作に入ったら，InSavingプロパティを書き換えなければならない．

ウィンドウのデザイン

では，実際の作業に入ろう．

まず，インストール可能なスクリーンセーバーの本体となるプログラムを作成する．ファイル名はVSSAVE.EXE，名前をV-S-SAVERとする．このプログラムから，スクリーンセーバーの動作を行うプログラムを起動させる．したがって，このプログラムの中に，スクリーンセーブ動作そのもののルーチンは含まれていないが，スクリーンセーバーを起動するタイミングについては，このプログラムの中で調べている．

まず，VBを起動する．すでに起動されている場合は，「File/New」で新しいプロジェクトを開く．

このまっさらなフォームの上に自由にコントロールを配置できるわけだが，焦ってはいけない．

ここでは，考えなければならない問題が，2つある．

まず，カスタムコントロールをどうやって取り込むかである．これは難しくない．「File/Add File...」メニューを選んで，vsaver1.vbxを指定すれば，プロジェクトウィンドウに，このファイルが追加され，ツールボックスに見慣れないアイコンが追加されたことがわかるだろう．

2番目の問題は，少々ややこしい．これは，コントロールの順番に関する問題である．[TAB]キーを押せば，フォーカスが次のコントロールに移る．また，スタティックテキストに設定されたアクセラレータキーを押せば，そのスタティックテキストの次にフォーカスを持てるコントロールに制御が移る．これを正確に理解していないと，キーボード操作のやりにくいプログラムになってしまう．コントロールの順番は，非常に大切である．

さて，デザイン中に，この順番を変える方法だが，私はうまい方法を知らない．CUT&PASTEすれば，特定のコントロールの順番を一番最後に移動させることはできると思うが．誰か，うまい方法を知っていたら教えて欲しい．

さて，だいたい内容を決めたら，[TAB]キーでフォーカスが動くべき順番にコントロールを置いていく．自分自身にアクセラレータ文字を持てるボタンなどはよいのだが，リストボックスなどはアクセラレータ文字を持てないので，必然的にスタティックテキストを添えることになる．このとき必ず，

(1)スタティックテキスト（アクセラレータ文字付き）

(2)リストボックスなど

の順番で置く．この間に，別のものも置かないほうが望ましい．

さて，アクセラレータ文字を持たないスタティックテキストやピクチャーボックスについては，いつ置いても関係ない．だから，ちょっとした説明文を追加したいな，などと思ったときには，いつでもスタティックテキストを置くことができる．実際，ここで解説するV-S-SAVERでも，バージョン番号やコピーライトの表記は，あとから思い立って入れたものである．

さて，それでは実際のV-S-SAVERのデザインにあたって，実際にやったこと，注意すべき点などを説明する．

まず，フォントについて．

通常，日本語Windowsであっても，いくつかの英語フォントが付いている．少なくとも，どのメーカーのWindows 3.0でも，Helv，Tms Roman，Roman，Modern，Script，Symbol（これは英語ではないが）を装備している．これを使わないのは損である．Helvは，線が太く，小さな文字でもはっきり見えるメリットがある．Tms Romanは，線端に飾りが付いていて，タイトルなどに見栄えがよい．どちらも，プロポーショナルだから，文字によって幅が違うので，並べたときの美しさは，並みの日本語フォントの比ではない．

今回は，表示内容が決まっているスタティックテキストで，フォントに凝ってみた．まず，基本線は，Helvである．VB自身のデフォルトもHelvである．もっとも見やすいフォントである．そして，題字だけは，Tms Roman

Windows プログラミング用語集
EMS（Expanded memory specification）　ロータス・インテル・マイクロソフトの3社によって提唱された拡張メモリの仕様

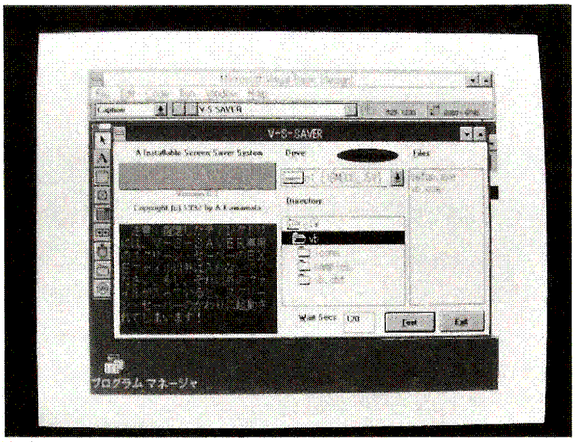


写真1　V-S-SAVERのデザイン

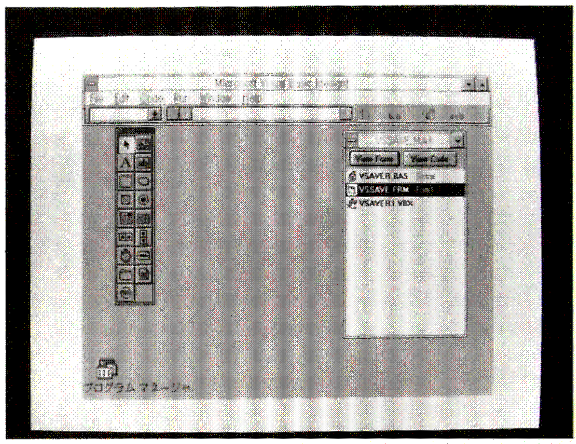


写真2　V-S-SAVERのプロジェクトとツールボックス

のもっとも大きなサイズを使用した．

さて，プロポーショナルなフォントを使うと文字列の長さがスパッと決まらない．そのため，スタティックテキストを置くべき位置に苦労することがある．そういう時は，左右に多めの領域を取って，Alignmentプロパティを Centerに設定(つまりセンタリングである)してしまうと，ぴたっと決められることがあるので，覚えておくとよい．

さて，それから，注意書きは日本語で表示したかったので，システムフォントに設定した．どのメーカーの日本語Windowsでも必ず入っている日本語フォントは，Systemだけと考えたほうがよい．そのため，泣く泣くSystemを使っている．

ここでの問題は，VB自身の入力フィールドがHelvで表示されていて，ごていねいにかな漢が未確定文字列をHelvで表示してしまうことだろう．英語フォントであるため，日本語の文字が化ける．長い日本語文字列を入力するのがめんどうくさい．英語フォントを全部外してWindowsを再起動するという荒業もあるが，ここは，メモ帳あたりで入力してから，CUT&PASTEで流し込むのが正解であろう．

それから，ドライブ，ディレクトリ，ファイルリストも，システムフォントを指定している．これは，ボリューム名，ディレクトリ名，ファイル名にカナや漢字が使われている可能性があるためだ．なお，現在のVBは英語版であるため，漢字のファイル名やディレクトリ名は正しく扱えない．したがって，漢字のディレクトリなどを指定したら，その後の動作はわからなくなる．しかし，単に表示するぐらいは考慮しておいたほうが，よいと思う．

スクリーンセーバーを起動するまでの時間を設定するエディットコントロールだけは，Helvのままにしてある．ここでは，1バイト文字の数字しか受け付けないので，「こんなもんでいいかな？」と思ってそうしてあるのだが，はっきりいって手抜きである．

さて，ここまでやっても，ウィンドウの見た目はかなり単調である．VBならではのお化粧をしてみよう．カラーの活用である．スタティックテキストには，バックカラーとフォアカラーを簡単に指定できる．わざわざ，ForeColorプロパティやBackColorプロパティを開くまでもない．Color Paletteのウィンドウを Windowメニューから開き，フォーム上のコントロールを選択してから色をクリックすればその色に変わる．フォアカラーとバックカラーは，Color Paletteウィンドウの脇のカラーサンプルの中央と脇をクリックすると切り替えられる．試行錯誤してみれば，すぐにわかるだろう．

ここで問題なのは，パレットに並んでいるすべての色が，文字に付けられないということである．文字のフォアカラーとバックカラーは，ソリッドカラーといって，システムがハード的に表示できる色しか利用できない．異なる色を並べて擬似的に中間色を出すような方法（ディザないしタイリングと呼ばれる）を受け付けない．そのため，コントロール全体のバックをディザで塗り潰しながら，文字のバックだけソリッドカラーで塗り潰すため，色にくい違いがでる可能性がある．

そういうトラブルを回避するために，なるべく，色数の少ない16色のシステムでカラーをデザインしておくことをお勧めする．

さて，トラブルといえば，もう1つやっかいな問題がある．

デザインする時のディスプレイドライバの解像度の問題である．

ご存じのことと思うが，VBでは，基本的にTWIPS（1/1440インチ）を単位にしてデザインを行う．そのため，

Windowsプログラミング用語集
GUI（Graphic user interface）　従来のコマンドベース・テキストベースのユーザーインターフェースに対してグラフィックを利用し視覚的に操作する方法.

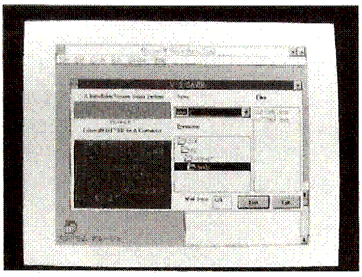


写真３　V-S-SAVER 実行中．Screen Saver コントロールは見えない

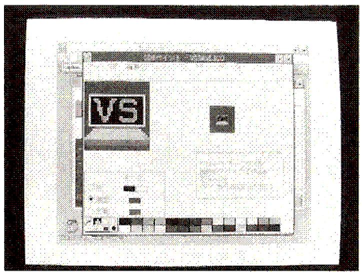


写真４　V-S-SAVER のアイコン

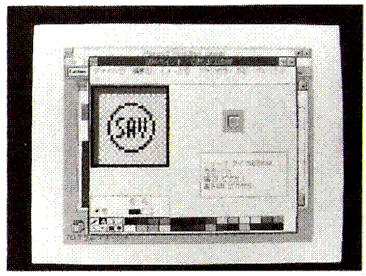


写真５　circlecd.bmp のデザイン

実際の画面上のピクセル数と，きっちり対応しないことがある．そういう場合，異なる解像度のディスプレイドライバを使うと，座標の設定内容に微妙な違いが出てしまう．

これは，もう，何を基準にするかの問題になるのだが，とりあえず，VGA を基準に考えておけばよいと思う．VGAの640×480のドライバは96 dpi×96dpiを想定して作られている．なるべく，この解像度でデザインしておくことをお勧めする．なお，PC-9801のノーマル解像度の640×400モードも，96dpi×96dpiなので，これは安心して使える．SuperVGAの1024×768でデザインするよりも，98の400ラインのほうがベターである．

さて，あとは細かいことであるが，ScreenSaverのコントロールは，Visibleプロパティをfalseにしておく．こうすると，デザイン時には見えるが，実行したときには見えなくなる．そのため，このコントロールは，フォームの上のどこに置いても関係ない．

また，フォームからは，最大化ボタンを取っておく．ウィンドウの中のデザインが決まっている以上，最大化は無意味である．MaxButtonプロパティをFalseにしておく．それから，アイコンも作っておき，これをIconプロパティに設定しておこう．

あとは，忘れずにCaptionに名前を入れておく．

カナアクセラレータ文字

さて，日本語Windowsには，アルファベットとカナの２種類のアクセラレータ文字があり，コントロールパネルで切り替えられる．

今回のプログラムでは，手抜きによってサポートしていないが，英語版のVBでもこれをサポートする方法がある．それを解説しよう．

まず，英語版のアクセラレータ文字とアルファベットのアクセラレータはまったく別の存在である．つまり，日本語Windowsでは，３種類のアクセラレータ文字が存在する．

つまり，表１のようになる．

英語版の通常の方法で，&を使って定義したものは，どのモードでも有効である．

問題は，日本語Windowsのアルファベットとカナのアクセラレータの前置記号をどうやって入力するかである．おわかりの方にはわかるとおり，¥036や¥037というのは，８進定数である．VBでは，これをそのまま入力できない．

結論からいえば，デザイン時に，これらのアクセラレータを設定するのは，絶望的である．

しかし，諦めるのは早い．プログラムで設定してしまうという荒業があるのだ．

たとえば，あるボタンに，アルファベットモードで「A」，カナモードで「ア」という文字をアクセラレータに設定したいとしよう．

ずばり，リスト１のような１文を実行すればOKである．&036などは，

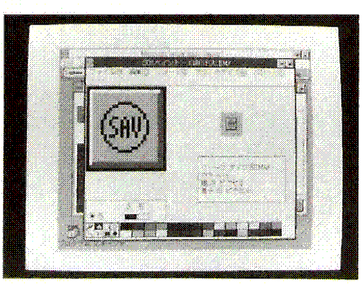


写真６　circlecu.bmp のデザイン

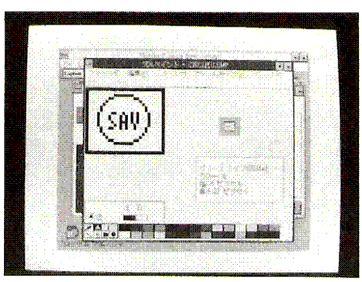


写真７　circleeu.bmp のデザイン

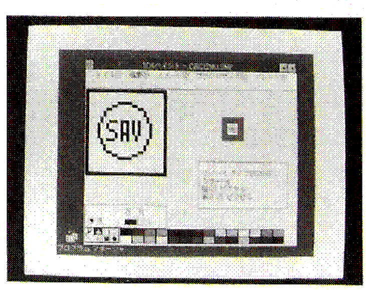


写真８　circlemu.bmp のデザイン

Windows プログラミング用語集
HMA（High memory area）　10000Hから約64Kbyteの領域のこと．A20アドレスラインを有効にするとリアルモードからも利用できる．

表1 アクセラレータ文字の種類一覧

	英語版	アルファベット	カナ
前に置く記号	&	¥036(&036)	¥037(&037)
アルファベットモード	有　効	有　効	無　効
カナモード	有　効	無　効	有　効

リスト1

```
Button1.Caption = "ラベル("+chr$(&036)+"A"+chr$(&037)+"P)"
```

VBでの8進定数の表記である．

これを実行するタイミングであるが，フォームのLoadイベントあたりが最適であろう．

汎用プロシージャの作成

さて，V-S-SAVERの外見はできた．これに，各コントロールのためのルーチンを書いていけばよいのだが，どのコントロールにも該当しないグローバルな定義や，共通に利用されるプロシージャなどは，別途作らなければならない．

まず，グローバル定義であるが，ここでは，WriteProfileStringとGetProfileStringとShowCursorをDeclareによって宣言している．これらは，WindowsのAPIである．WindowsのAPIといっても，正体はただのDLLであるから，Declareで問題なく宣言できるのである．

さて，詳細はSDKのマニュアルなどを見ていただくとして，おおまかな機能を説明しておく．

WriteProfileStringは，win.iniに，項目を書き込む（あるいは置き換える）機能を持ったAPIである．モジュールの入っているディレクトリを記録しておくことと，セーブ動作に入るまでの秒数を記録しておくために使う．

GetProfileStringは，その逆に，win.iniに書かれた項目を1つ読み出す機能を持っている．もし，指定項目が書かれていなければ，デフォルト値が使用される．

ShowCursorは，マウスポインタ（カーソル）の表示，非表示を切り替えるAPIである．スクリーンセーバーの動作時には，マウスポインタは消しておく必要がある．マウスポインタが画面に焼き付いてしまっては，話にならないからである．それもコントロールしなければならない．

さて，もう1つ，どのコントロールにも属さない汎用的なルーチンも書いておこう．

これは，global.basではなく，フォームの中に書き込むことになる．まず，フォームのコード編集ウィンドウを開いて欲しい．その中の，Objectのコンボボックスから，(general)という項目を選んで欲しい．ここには，どのコントロールにも属さないプロシージャを記述できる．Procのコンボボックスが(declararions)（新規プロシージャの定義）であることを確認して，おもむろに，

Sub ExecSaver[リターン]

と入力すると，自動的に最後にEnd Subが添えられる．

この，SubとEnd Subの間に，次々とプログラムを入力していけばよいのである．

まず，指定ファイル名のセーバーモジュールを起動するプロシージャを作ってみよう．

モジュールといっても，ただのVBプログラムをモジュールとして使うつもりなので，SHELL関数で引数で渡されたファイル名のEXEファイルを起動すればよい．

ただし，注意すべき点が2つある．

まず，カレントディレクトリの問題である．ドライブ，ディレクトリのコントロールで示したディレクトリがカレントになっていない可能性があるので，これを強制的にカレントディレクトリに切り替えなければならない．

そのためにリスト2のような2行を添える．

それから，起動されるべきスクリーンセーバーは，フルスクリーン動作でなければ意味がない．また，フォーカスをしっかり持っていないと，キー入力・マウス入力を受け取れなくて，終了のタイミングがわからない可能性が

リスト2

```
ChDrive Drive1.Drive
ChDir Dir1.Path
```


Windowsプログラミング用語集
VCPI (Virtual control program interface)　386以上でDOSエクステンダや仮想86モードなどを使ったアプリケーションからプロテクトメモリを利用するためのI/F.

リスト 3

```
Sub ExecSaver (fname As String)
    ChDrive Drive1.Drive
    ChDir Dir1.Path
    dummy = Shell(fname, 3)
End Sub
```

リスト 4

```
ExecSaver (File1.List(Int(File1.ListCount * Rnd(1))))
```

ある．そのため，Shell関数の第2引数は，3，つまり「最大化かつフォーカスを持つ」でなければならない．

以上を総合すると，リスト3のように書ける．

もう1つ，ランダムにモジュールを選んで起動するプロシージャも作ってみよう．

候補は，すでにファイルリストボックスに上がっているので，この中からランダムに選ぶだけでよい．すでに作ったExecSaverを使えば，リスト4の1行で済んでしまう．ちなみに，File1には，“*.exe”というマスクがかかっているので，EXE以外のファイルを選んでしまう危険はない．しかし，セーバーモジュールでないEXEファイルを選んでしまう危険はある．

さて，これで問題が出るのは，1個もモジュールがないときで，それを回避するためにリスト5のようなコードも入れておく．メッセージが英語なのは，一種の手抜きである．

コントロールのプロシージャの作成

さて，順番にコントロールに書き込んだイベント・プロシージャの内容を見てみよう．

Testボタン

選択されているモジュールをその場で起動してみる機能である．だから，Clickイベントのときに，選択されているファイル名を引数にしてExec-Saverを呼べばよいのだが，ファイルが選択されていない場合を想定して，チェックを1つ入れておく必要があるだろう．リスト6のように書いておく．

Exitボタン

終了ボタンである．単純に，Clickイベントの時にendを実行させればよいように思えるが，こうするとフォームのUnloadイベントが走らないで終了してしまうようである．Unloadイベントでは，ディレクトリとセーブ動作に入るまでに時間が記録されるので，これが走らないのはまずい．そのため，一度

```
Unload Form1
```

を実行してから

```
end
```

を走らせている．ほかの方法で終了させるとUnloadが走るので，バグか仕様かよくわからない．

ドライブボックス

Changeイベントに対応して，ディレクトリリストに変更を通知するだけでよい．

ただ，メディアの入っていないドライブなどにアクセスしたときにエラーで止まってしまう可能性があるので，少々のごまかしを行う．

リスト7のようにすることで，エラーが起こってもそ知らぬ顔で処理を継続させてしまっている．これは，一種の手抜きである．

ディレクトリリスト

これも，Changeイベントに対応して，ファイルリストのほうに通知してやれば済む．これも，同様に，リスト8のようにしてある．

ファイルリスト

これが存在している理由は，このリストそのものが何かのアクションを起こすことではなく，ユーザーにモジュールの一覧表を見せることと，ランダ

リスト 5

```
If File1.ListCount = 0 Then
    MsgBox "No Savers!, Can't execute any screen saver"
    Exit Sub
End If
```

リスト 6

```
If File1.FileName <> "" Then
        ExecSaver (File1.FileName)
End If
```

リスト 7

```
On Error Resume Next
Dir1.Path = Drive1.Drive
On Error GoTo 0
```

リスト 8

```
On Error Resume Next
File1.Path = Dir1.Path
On Error GoTo 0
```

ムに選択してモジュールを起動するときに，データベースとして利用するためである．したがって，とくにイベントを記述する必要はない．

ただ，サービスということで，あるモジュールのファイル名をダブルクリックすると，そのモジュールが即起動して，スクリーンセーブ動作に入るという機能は付けてある．そのためDblClickイベントにTestボタンと同じ内容を入れてある．

フォーム

さて，かんじんのフォームであるが，いくつかの処理を行っている．

まず，Loadイベントと，Unloadイベントで，デフォルトのディレクトリと，セーブ動作に入る秒数のパラメータをロード・セーブしている．

さて，ここで問題になるのは，WindowsAPIへのアクセスである．ものによっては，普通のプロシージャのように使うことはできないのである．ここで使っているGetProfileStringはその典型である．

ここでは，リスト9のように使っている．

まず，受け取る文字列を入れる領域を，「確保」しなければならない．Basicでの文字列は，自動的に文字列の長さのぶんだけサイズが変化するのだが，Cの文字列は必ず最初に最大サイズを決めなければならない．WindowsのAPIはCのルールで扱うから，同じ制約を受ける．

したがって，中身はなくても，長さだけは取ってある文字列をstring$で作って，これを変数ret$に入れておく．そして，この変数ret$を受け取り用変数に指定してAPIを呼び出す．

もう1つ注意することは，APIの中でエラーが起こった場合，エラーメッセージを表示して実行を打ち切ってくれることはない，ということだ．APIのエラーはすべてリターン値で示される．だから，必ずリターン値を調べ，ちゃんと実行されたかどうかを確認しなければならない．

なお，UnloadイベントでのWriteProfileStringでは，とくにリターン値をチェックしていないが，これは単なる手抜きである．

フォームは，もう1つ，DDEのためのイベントも準備しなければならない．セーバーモジュールと，V-S-SAVERの間の情報交換にDDEを使っているためだ．LinkExecuteのイベントに対応して，リスト10のようなルーチンを書く．

DDEの目的は，セーバーモジュールがセーブ動作を始めたことと，終了したことを通知することである．したがって，引数の文字列には，"0"と"1"しか来ないことになっている．この文字列をval関数で数値に直してScreenSaver1のInSavingプロパティに代入すればおしまいである．InSavingプロパティが1であれば，StartSaverイベントを起こさないようになっている．

同時に，もう1つ，マウスポインタ（カーソル）の表示，非表示の制御も行っている．これは，セーバーモジュール側でやってもよいのだが，なるべくセーバーモジュールの開発を簡単にするために，V-S-SAVER側でやっ

リスト 9

```
ret$ = String$(255, 0)
success% = GetProfileString("VSSAVE", "MODULEPATH", "", ret$, 255)
If success% <> 0 Then
    Drive1.Drive = ret$
    Dir1.Path = ret$
End If
```

リスト 10

```
Sub Form_LinkExecute (CmdStr As String, Cancel As Integer)
    ScreenSaver1.InSaving = Val(CmdStr)
    If Val(CmdStr) = 0 Then
        ShowCursor 1
    Else
        ShowCursor 0
    End If
    Cancel = 0
End Sub
```


Windows プログラミング用語集
XMS（Extended memory specification）　80286以上のCPUで，10000H以上のアドレスにマッピングされるメモリの管理仕様

てしまうことにした．ShowCursorの引数が0なら非表示に，1なら表示状態に切り替える．

最後に，Cancel =0によって，LinkExecuteの動作が成功したことを示さなければならない．

リスト 11

```
If Val(Text1.Text) <= 5 Then
    Beep
    Exit Sub
End If
```

スクリーンセーバー

カスタムコントロールのScreenSaverであるが，必要なイベントはただ1つ．StartSaverである．

今回は，単純に，すでに説明したExecRandomSaverを呼び出すだけにしてあるが，もっと凝った選択方法に変更するのも簡単だろう．

秒数入力テキスト

セーブ動作に入るまでの秒数を入力するエディットコントロールである．これは，Changeイベントが発生したら，その値をScreenSaver1のWEaitSeconsプロパティに代入すればよい．

ただ，まともな内容ではないものが入力される危険があるので，リスト11のようなチェック機構を設けている．もし，内容が数字でなければVal関数の値は0になり，エラーに引っ掛かる．とりあえず，5秒よりも小さい値は入力できない仕様である．

さて，これで，V-S-SAVERの説明はすべて終わりである．多少，プロパティの設定で，説明を省いたものもあるが，これでおしまいである．かなり，あっけないものだ．

それでも，ちゃんと動くのだから，たいしたものである．

あとは，実行してみて問題がなければ，「File/Make EXE file...」からEXEファイルを作る．これで作業は完了である．EXEファイルを配布するときは，VSAVER1.VBXとVBRUN100.DLLを添える．

モジュールの作り方

さて，V-S-SAVERを使うには，セーブ動作を行うモジュールが必要である．

リスト 12　VSSAVE プロパティ一覧

オブジェクト	プロパティ	設定値
Form1	AutoRedraw	False
	BackColor	&H80000005&
	BorderStyle	1-Fixed Single
	Caption	V-S-SAVER
	ControlBox	True
	DrawMode	13-Copy Pen
	DrawStyle	0-Solid
	DrawWidth	1
	Enabled	True
	FillColor	&H00000000&
	FillStyle	1-Transparent
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikethru	False
	FontTransparent	True
	FontUnderline	False
	ForeColor	&H80000005&
	FormName	Form1
	Height	4740
	Icon	(Icon)
	Left	525
	LinkMode	1-Sever
	LinkTopic	Form1
	MaxButton	False
	MinButton	True
	MousePointer	0-Default
	Picture	(none)
	ScaleHeight	4290
	ScaleLeft	0
	ScaleMode	1-Twip
	ScaleTop	0
	ScaleWidth	8760
	Tag	
	Top	1230
	Visible	True
	Width	8880
	WindowState	0-Normal
Label1①	Alignment	2-Center
	AutoSize	False
	BackColor	&H80000005&
	BorderStyle	0-None
	Caption	A Installable Screen Saver System
	CtlName	Label1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H00000080&
	Height	315
	Index	
	Left	0
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	0
	Tag	
	Top	150
	Visible	True
	Width	3975
Label2②	Alignment	2-Center
	AutoSize	False
	BackColor	&H00FF0000&
	BorderStyle	1-Fixed Single
	Caption	V-S-SAVER
	CtlName	Label2
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Tms Rmn
	FontSize	24
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H0000FFFF&
	Height	615
	Index	
	Left	120
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	1
	Tag	
	Top	450
	Visible	True

Windows プログラミング用語集

アイコン（Icons）　プログラムの種類や状態，処理，警告などをユーザーに視覚的に伝える小さなシンボルマーク．

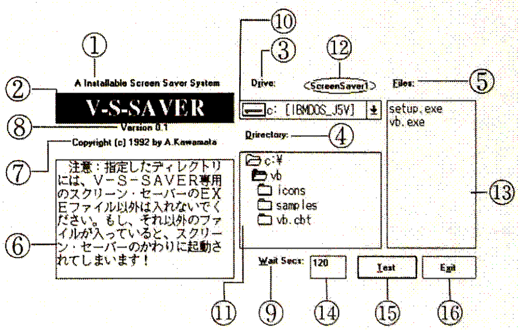
	Width	3615
Label3③	Alignment	0-Left Justify
	AutoSize	False
	BackColor	&H00FF0000&
	BorderStyle	1-Fixed Single
	Caption	D&rive
	CtlName	Label3
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	315
	Index	
	Left	4080
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	2
	Tag	
	Top	150
	Visible	True
	Width	975
Label4④	Alignment	0-Left Justify
	AutoSize	False
	BackColor	&H80000005&
	BorderStyle	0-None
	Caption	&Drirectory
	CtlName	Label4
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	315
	Index	
	Left	3960
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	4
	Tag	
	Top	1200
	Visible	True
	Width	1335
Label5⑤	Alignment	0-Left Justify
	AutoSize	False
	BackColor	&H80000005&
	BorderStyle	0-None
	Caption	&Files:
	CtlName	Label5
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	315
	Index	
	Left	6960
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	6
	Tag	
	Top	150
	Visible	True
	Width	1575
Label6⑥	Alignment	0-Left Justify
	AutoSize	False
	BackColor	&H000000FF&
	BorderStyle	1-Fixed Single
	Caption	注意：指定したディレクトリには、V－S－SAVER専用のスクリーン・セーバーのEXEファイル以外は入れないでください。もし、それ以外のファイルが入っていると、スクリーン・セーバーのかわりに起動されてしまいます！
	CtlName	Label6
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	False
	FontItalic	False
	FontName	System
	FontSize	13.5
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H0000FFFF&
	Height	2415
	Index	
	Left	120
	LinkItem	

	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	10
	Tag	
	Top	1800
	Visible	True
	Width	3615
Label7⑦	Alignment	2-Center
	AutoSize	False
	BackColor	&H80000005&
	BorderStyle	0-None
	Caption	Copyright(C)1992 by A.Kawamata
	CtlName	Label7
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H00008000&
	Height	315
	Index	
	Left	0
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	11
	Tag	
	Top	1350
	Visible	True
	Width	3855
Label8⑧	Alignment	2-Center
	AutoSize	False
	BackColor	&H00FFFFFF&
	BorderStyle	0-None
	Caption	Version 0.1
	CtlName	Label8
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H00808080&
	Height	315
	Index	
	Left	0
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	13
	Tag	
	Top	1050
	Visible	True
	Width	3855
Label9⑨	Alignment	1-Right Justify
	AutoSize	False
	BackColor	&H80000005&
	BorderStyle	0-None
	Caption	&Wait Secs:
	CtlName	Label9
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	315
	Index	
	Left	3840
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	TabIndex	14
	Tag	
	Top	3750
	Visible	True
	Width	1335
Drive1⑩	BackColor	&H80000005&
	CtlName	Drive1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	False
	FontItalic	False
	FontName	System
	FontSize	13.5
	FontStrikethru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	390
	Index	
	Left	3840
	MovePointer	0-Default
	TabIndex	3
	TabStop	True
	Top	600

Windows プログラミング用語集

アイソトロピックモード（IsoTropic mode）　Windowsのマッピングモードのひとつで，ウィンドウのサイズ変更時に画像比を維持しサイズの変更ができる。

	Visible	True
	Width	2895
Dir1⑪	BackColor	&H80000005&
	CtlName	Dir1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	False
	FontItalic	False
	FontName	System
	FontSize	13.5
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	1965
	Index	
	Left	3840
	MousePointer	0-Default
	TabIndex	5
	TabStop	True
	Tag	
	Top	1650
	Visible	True
	Width	2895
ScreenSaver1⑫	BackColor	&H000000FF&
	BoderStyle	0-None
	Caption	ScreenSaver1
	CircleShape	1-Oval
	CtlName	ScreenSaver1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FlashColor	&H00000000&
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikethru	False
	FontUnderline	False
	Height	315
	Index	
	Insaving	0-NotInSaving
	Left	5160
	MousePointer	0-Default
	SampleRate	0
	TabIndex	12
	TabStop	True
	Tag	
	Top	150
	Visible	False
	WaitSeconds	120
	Width	1455
File1⑬	Archive	True
	BackColor	&H80000005&
	CtlName	File1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	False
	FontItalic	False
	FontName	System
	FontSize	13.5
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	3000
	Hidden	False
	Index	
	Left	6840
	MousePointer	0-Default
	Normal	True
	Pattern	*.exe
	ReadOnly	True
	System	False
	TabIndex	7
	TabStop	True
	Tag	
	Top	600
	Visible	True
	Width	1815
Text1⑭	BackColor	&H80000005&
	BorderStyle	1-Fixed Single
	CtlName	Text1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	465
	Index	
	Left	5280
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	MultiLine	False
	ScrollBars	0-None
	TabIndex	15
	TabStop	True
	Tag	
	Text	120
	Top	3750
	Visible	True
	Width	735
Command1⑮	BackColor	&H80000005&
	Cancel	False
	Caption	&Test
	CtlName	Command1
	Default	True
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	Height	465
	Index	
	Left	6240
	MousePointer	0-Default
	TabIndex	8
	TabStop	True
	Tag	
	Top	3750
	Visible	True
	Width	1215
Command2⑯	BackColor	&H80000005&
	Cancel	False
	Caption	E&xit
	CtlName	Command2
	Default	False
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	Height	465
	Index	
	Left	7560
	MousePointer	0-Default
	TabIndex	9
	TabStop	True
	Tag	
	Top	3750
	Visible	True
	Width	1095



リスト 13　VSSAVE ソース

```
Sub Command2_Click ()
    Unload Form1
    End
End Sub

Sub Drive1_Change ()
    On Error Resume Next
    Dir1.Path = Drive1.Drive
    On Error GoTo 0
End Sub

Sub Dir1_Change ()
    On Error Resume Next
    File1.Path = Dir1.Path
    On Error GoTo 0
End Sub

Sub ExecSaver (fname As String)
    ChDrive Drive1.Drive
    ChDir Dir1.Path
    dummy = Shell(fname, 3)
End Sub

Sub File1_DblClick ()
```

アクセラレータキー（Accelerator key）　短縮キーのことで，任意のキーストロークによってマウスのクリックなどと同じようにコマンドメッセージを生成する。

```
    If File1.FileName <> "" Then
        ExecSaver (File1.FileName)
    End If
End Sub

Sub Command1_Click ()
    If File1.FileName <> "" Then
        ExecSaver (File1.FileName)
    End If
End Sub

Sub Form_Load ()
    ret$ = String$(255, 0)
    success% = GetProfileString("VSSAVE", "MODULEPATH", "", ret$, 255)
    If success% <> 0 Then
        Drive1.Drive = ret$
        Dir1.Path = ret$
    End If

    ret$ = String$(255, 0)
    success% = GetProfileString("VSSAVE", "WAITSECS", "", ret$, 255)
    If success% <> 0 Then
        Text1.Text = ret$
        ScreenSaver1.WaitSeconds = Val(ret$)
    End If
End Sub

Sub Form_Unload (Cancel As Integer)
    WriteProfileString "VSSave", "ModulePath", (File1.Path)
    WriteProfileString "VSSave", "WaitSecs", (Text1.Text)
End Sub

Sub Form_LinkExecute (CmdStr As String, Cancel As Integer)
    ScreenSaver1.InSaving = Val(CmdStr)
    If Val(CmdStr) = 0 Then
        ShowCursor 1
    Else
        ShowCursor 0
    End If
    Cancel = 0
End Sub

Sub ExecRandomSaver ()
    If File1.ListCount = 0 Then
        MsgBox "No Savers!, Can't execute any screen saver"
        Exit Sub
    End If
    ExecSaver (File1.List(Int(File1.ListCount * Rnd(1))))
End Sub

Sub ScreenSaver1_StartSaver ()
    ExecRandomSaver
End Sub

Sub Text1_Change ()
    If Val(Text1.Text) <= 5 Then
        Beep
        Exit Sub
        End If
        ScreenSaver1.WaitSeconds = Val(Text1.Text)
    End Sub
```

これの作り方を説明しよう．

もっとも基本になるのは，VSBLANK.EXEという，ただ画面を真っ黒にするだけのモジュールである．ただ単に，VBで書いた普通のプログラムである．実際にモジュールを作るときには，このVSBLANKを読み込んで修正を加えて作れば楽であろう．もし，そういう作業をしてみたいという人は，VSBLANKの内容に関する説明を読み飛ばしてもかまわない．

まず，VSBLANKのフォームは，完全に真っ黒なウィンドウでなければならない．そのため，フォームのプロパティのCaption，ControlBox，MaxButton，MinButtonを取り除き，BorderStyleを0 - Noneに設定する．そして，BackColorを黒に設定すれば完全である．デザイン時にはとくに変化はないが，これを実行させると真っ黒なウィンドウになる．

さて，このフォームにリスト14の2つのプロシージャを追加する．どちらも，特定のコントロールではなく，(general)に追加する．

見ての通り，セーバーのスタートと終了をV-S-SAVERに通知するためのルーチンである．ExitSaverのほうは，最後にendを入れてあり，ここで終了することになる．

さて，通知するためのDDEであるが，LinkExecuteを使う限り，難しいものではない．LinkTopicプロパティ，LinkItemプロパティを設定してから，LinkModeプロパティでチャンネルを設定し，その後に，LinkExecuteメソッドを実行すればよいだけである．受け取るほうは，LinkExecuteイベントに対応するルーチンを書くだけでよい．

ただ，問題は，DDEを受ける側はフォームでよいのだが，DDEを仕掛ける側はフォームではできないのである．そのため，DDEを仕掛けるためだけに，Text1というテキストボックスを

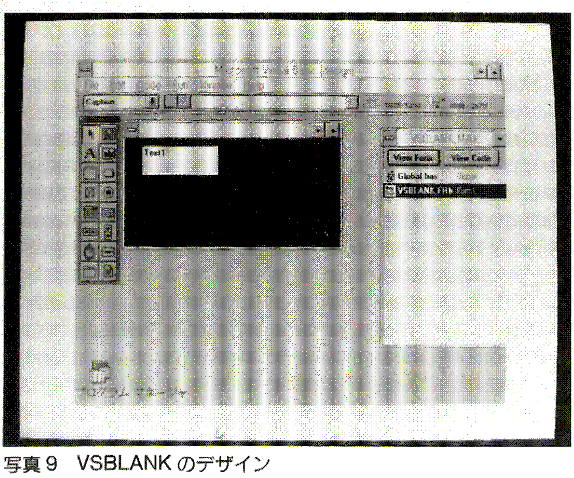


写真9　VSBLANKのデザイン

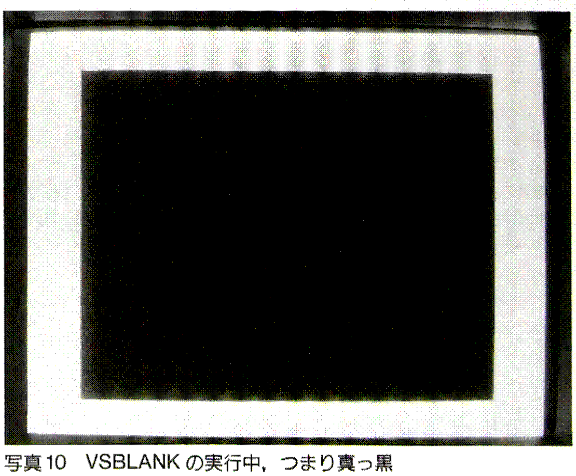


写真10　VSBLANKの実行中，つまり真っ黒

Windows プログラミング用語集
アトム（Atoms）　Windowsはアトムテーブルと呼ばれる文字列領域をもち，アトムはこの文字列領域の文字列を示す整数値である．

作ってある．これはVisibleプロパティがFalseなので，実行すれば見えなくなる．しかし，消えてしまったわけではないので，DDEの機能は，きちんと実行してくれる．

そして，真っ黒なフォームに，以下のイベントを設定する．

Loadイベント→EnterSaverプロシージャの呼び出し

KeyDown，KeyUp，MouseDown，MouseUp，MouseMoveの各イベント→ExitSaveプロシージャの呼び出し

つまり，このウィンドウがアクティブなときに，キーボードかマウスで何かをやろうとしたら，即終了してしまうのである．

ちなみに，ClickやKeyPressのイベントでは十分でない．たとえば，タスクを切り替えるために，ユーザーは

リスト 14

```
Sub EnterSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "1"
    Text1.LinkMode = 0
End Sub

Sub ExitSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "0"
    Text1.LinkMode = 0
    End
End Sub
```

リスト 16 VSBLANK ソース

```
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)
    ExitSaver
End Sub

Sub Form_KeyUp (KeyCode As Integer, Shift As Integer)
    ExitSaver
End Sub

Sub Form_MouseDown (Button As Integer, Shift As Integer, x As Single, y As Single)
    ExitSaver
End Sub

Sub Form_MouseMove (Button As Integer, Shift As Integer, x As Single, y As Single)
    Static First As Integer, oldx As Single, oldy As Single
    If First = 0 Then
        First = 1
    Else
        If oldx <> x Or oldy <> y Then
            ExitSaver
        End If
    End If
    oldx = x
    oldy = y
End Sub

Sub Form_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
    ExitSaver
End Sub

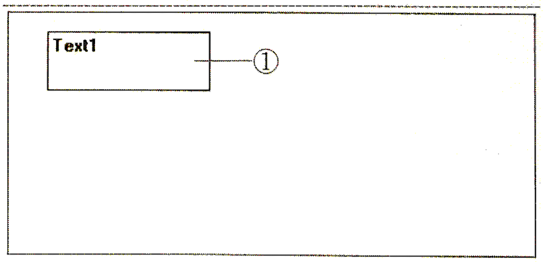
Sub ExitSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "0"
    Text1.LinkMode = 0
    End
End Sub

Sub EnterSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "1"
    Text1.LinkMode = 0
End Sub

Sub Form_Load ()
    EnterSaver
End Sub
```

リスト 15 VSBLANK プロパティ一覧

オブジェクト	プロパティ	設定値
Form1	AutoRedraw	False
	BackColor	&H00000000&
	BorderStyle	0-None
	Caption	
	ControlBox	False
	DrawMode	13-Copy Pen
	DrawStyle	0-Solid
	DrawWidth	1
	Enabled	True
	FillColor	&H00000000&
	FillStyle	1-Transparent
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikethru	False
	FontTransparent	True
	FontUnderline	False
	ForeColor	&H00FFFFFF&
	FormName	Form1
	Height	2670
	Icon	(Icon)
	Left	1035
	LinkMode	1-Sever
	LinkTopic	Form1
	MaxButton	False
	MinButton	False
	MousePointer	4-Icon
	Picture	(none)
	ScaleHeight	2220
	ScaleLeft	0
	ScaleMode	1-Twip
	ScaleTop	0
	ScaleWidth	4725
	Tag	
	Top	1290
	Visible	True
	Width	4845
	WindowState	0-Normal
Text1①	BackColor	&H80000005&
	BorderStyle	1-Fixed Single
	CtlName	Text1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	615
	Index	
	Left	360
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	MultiLine	False
	ScrollBars	0-None
	TabIndex	0
	TabStop	True
	Tag	
	Text	Text1
	Top	150
	Visible	False
	Width	1695



[CTRL]キーを押しながら[TAB]キーを押すかもしれない．こういう場合，KeyDownであれば[CTRL]キーのダウンを受け取るのでセーバーを終了させるチャンスがあるが，KeyPressのイベントは発生しないため，セーバーの気がつかないうちにほかのウィンドウがアクティブになってしまう可能性がある．また，すべてのマウスについて起こるできごとは，最大化されたウィンドウだから受け取れるのであって，これが最大化されていないウィンドウなら，ウィンドウの外でマウスを動かしても，ボタンを押しても，イベントを受け取ることはできない．

さて，VSBLANK.EXEの説明は，これで終わりである．蛇足ながら，モジュールはDDEを行うため，RUNさせるときにはV-S-SAVERを起動しておかなければならない．

サンプルモジュール VSLINE1

さて，VSBLANK.EXEをベースにして，多少は動きのあるスクリーンセーバーを作ってみよう．

まず，VSBLANK.MAKを読み込んでから，「Save Project As...」でプロジェクトの名前を，「Save File AS...」でフォームの名前をVSLINE1に変更しておく．これをやっておかないと，VSBLANKに上書きしてしまうので注意が必要である．ついでに，「Make EXE File...」のダイアログを開いて，ApplicationのTitleも，VSLINE1に変更しておくこと．これは忘れやすいので，注意が必要である．

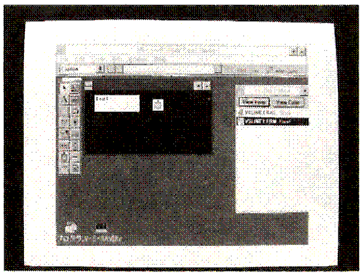


写真11　VSLINE1のデザイン

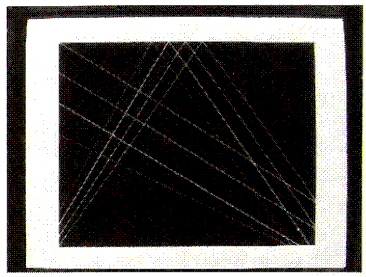


写真12　VSLINE1の実行中

さて，このセーバーモジュールは，色とりどりのラインが画面の中を動きまわるものとする．

厳密には，画面の4隅を周回する点を4個取り，この4点の間をラインで四角く結ぶ．色はランダムに決め，XORで描く．XORなら，2度描けば消えてくれるためだ．そして，過去4回分の座標を記憶させて，軌跡を残すようにする．色は，基本8色の中からランダムに選ぶものとする．変化を付けるため，移動する点の4つのうち，2つは逆廻りに廻すことにする．

さて，動く絵を実現するには，2種類の方法がある．

第1の方法は，タイマを使って1/10秒毎などのインターバルでイベントを起こさせ，そこで再描画する方法である．

第2の方法は，連続してVBのルーチンの中で処理をループさせ，DoEventsを呼ぶ方法である．頻繁にDoEventsを呼び出せば，ほかのアプリケーションにも実行のチャンスが与えられる．

今回は，第1の方法を使った．こちらのほうがオーソドックスであり，同時に，CPUの速度に関係なく，動きのスピードが一定であるためだ．

しかし，欠点もある．システムに15個しかないタイマを消費してしまうのである．ちなみに，VSAVER1.VBXもタイマを1個使うため，これが立ち上がっているときには2個のタイマを使ってしまうことになる．まあ，通常の使い方では問題ないとは思うが．

さて，軌跡を残すため，古い座標は保存されなければならない．また移動する量と方向も多少変化を付けたいので，これを保持する変数も取りたい．そのために，グローバルな変数の定義としてリスト17のようなものを宣言している．

リスト17

```
Global xpos(4, 4) As Single, ypos(4, 4) As Single
```

xは座標とyは座標を保持する．過去4回分の座標について，点が4点あるので，4×4の配列となる．

Global col(4) As Long

過去4回分の色を保持する．

Global speed(4) As Single

点の移動スピードである．多少乱数の要素を加えている．マイナスの値であれば，逆回りとなる．

Global dire(4) As Single

現在，画面のどの縁にいるのかを示す．0:上，1:右，2:下，3:左とする．

これらのグローバル変数は，フォームのLoadイベントで初期化している．

Windowsプログラミング用語集
イベント（Event）　「マウスがクリックされた」や「メニューを選択した」などのユーザー側の行為や，各プログラム自身から発せられる各種命令を指す．

さて，実際に描画するルーチンとして，以下のような２つのプロシージャを準備した．

まず，４点の間を，指定したカラーで結ぶルーチンである．変数のデータ型がSingle(!)であるのは，座標がTWIPSで渡されてくるからである．なお，描画のXOR指定については，デザイン時にフォームのDrawModeを7 - Xor Penに設定することでやっている．何も，すべてプログラムで書く必要はないのである．内容はリスト18のようになる．

もう１つは，

Sub DoNext ()

である．長くなるのでリスト20を見て欲しい．これは，タイマによる１ステップの処理を行う．具体的には，以下のステップで処理を行う．

いちばん古い線を消去する．
過去の座標とカラーを１つずつずらす．
新しい点の座標を計算する．
新しい色を決定する．
新しい座標について線を描画する．

このプロセスをタイマイベントが入るたびに呼び出していれば，アニメーションするわけである．

なお，タイマイベントのインターバルは，デザイン時にタイマのIntervalプロパティに100ミリ秒(つまり，1/10秒）を代入してある．

第１部の終わりに

VBとVSAVER1.VBXを手に入れたあなたは，Ｃもアセンブラも知らなくても，日曜日１日しか時間のない人でも，スクリーンセーバーを作れる

リスト 18

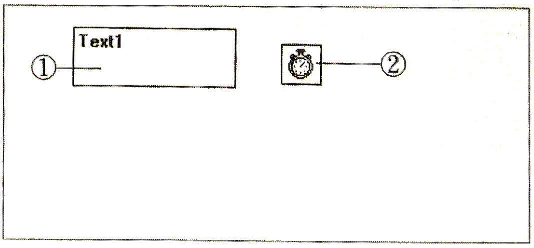
```
Sub DrawIt (x0!, y0!, x1!, y1!, x2!, y2!, x3!, y3!, c&)
    Line (x0!, y0!)-(x1!, y1!), c&
    Line -(x2!, y2!), c&
    Line -(x3!, y3!), c&
    Line -(x0!, y0!), c&
End Sub
```

リスト 19　VSLINE1.BAS

```
1: Global xpos(4, 4) As Single, ypos(4, 4) As Single
2: Global col(4) As Long
3: Global speed(4) As Single
4: Global dire(4) As Single
5:
```

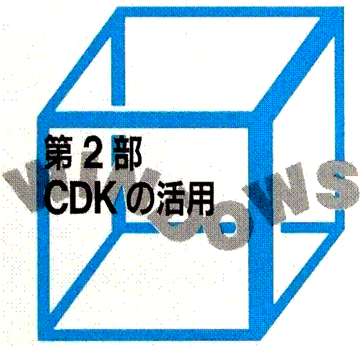
リスト 20　VSLINE1 プロパティ一覧

オブジェクト	プロパティ	設定値
Form1	AutoRedraw	False
	BackColor	&H00000000&
	BorderStyle	0-None
	Caption	
	ControlBox	False
	DrawMode	7-XorPen
	DrawStyle	0-Solid
	DrawWidth	1
	Enabled	True
	FillColor	&H00000000&
	FillStyle	1-Transparent
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikethru	False
	FontTransparent	True
	FontUnderline	False
	ForeColor	&H00FFFFFF&
	FormName	Form1
	Height	2670
	Icon	(Icon)
	Left	1035
	LinkMode	1-Sever
	LinkTopic	Form1
	MaxButton	False
	MinButton	False
	MousePointer	4-Icon
	Picture	(none)
	ScaleHeight	2220
	ScaleLeft	0
	ScaleMode	1-Twip
	ScaleTop	0
	ScaleWidth	4725
	Tag	
	Top	1290
	Visible	True
	Width	4845
	WindowState	0-Normal
Text1①	BackColor	&H80000005&
	BorderStyle	1-Fixed Single
	CtlName	Text1
	DragIcon	(none)
	DragMode	0-Manual
	Enabled	True
	FontBold	True
	FontItalic	False
	FontName	Helv
	FontSize	8.25
	FontStrikeThru	False
	FontUnderline	False
	ForeColor	&H80000008&
	Height	615
	Index	
	Left	360
	LinkItem	
	LinkMode	0-None
	LinkTimeout	50
	LinkTopic	
	MousePointer	0-Default
	MultiLine	False
	ScrollBars	0-None
	TabIndex	0
	TabStop	True
	Tag	
	Text	Text1
	Top	150
	Visible	False
	Width	1695
Timer1②	CtlName	Timer1
	Enabled	True
	Index	
	Interval	100
	Tag	



立場に立ったわけである．あとは，アイデアで勝負だ．指をくわえて，アフターダークあたりを見ている必要はない．みんなで，おもしろいスクリーンセーバーを作り出そう．

それから，本文中には，VBでアプリケーションを開発するために有用な情報を多数ちりばめたつもりである．これも，参考になれば幸いである．



第2部 CDKの活用

VBのカスタムコントロールとはどういうものであろうか．

今回のスクリーンセーバーカスタムコントロールにしても，V-S-SAVERだけでなく，いくらでもほかのプログラムに転用が可能である．これを使って，独自のスクリーンセーバーを開発することができる．しかし，そのためのCやSDKの知識は不要である．

カスタムコントロールは，Cやアセンブラのルーチンを VBから呼び出すための手段ではない．むしろ，VB自身の能力を拡大するためのツールであると考えるべきだろう．つまり，特定のアプリケーションに必要な機能をカスタムコントロールとして作るのではなく，汎用的なカスタムコントロールを集め，VB上で統合して新しいアプリケーションを開発するような発想をすべきだ．

今回は，その発想に乗っ取って，スクリーンセーバー機能を実現するカスタムコントロールを作ってみた．しかし，このカスタムコントロールは，ス

リスト 21 VSLINE1 ソース

```
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)
    ExitSaver
End Sub

Sub Form_KeyUp (KeyCode As Integer, Shift As Integer)
    ExitSaver
End Sub

Sub Form_MouseDown (Button As Integer, Shift As Integer, x As Single, y As Single)
    ExitSaver
End Sub

Sub Form_MouseMove (Button As Integer, Shift As Integer, x As Single, y As Single)
    Static First As Integer, oldx As Single, oldy As Single
    If First = 0 Then
        First = 1
    Else
        If oldx <> x Or oldy <> y Then
            ExitSaver
        End If
    End If
    oldx = x
    oldy = y
End Sub

Sub Form_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
    ExitSaver
End Sub

Sub ExitSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "0"
    Text1.LinkMode = 0
    End
End Sub

Sub EnterSaver ()
    Text1.LinkTopic = "VSSAVE|Form1"
    Text1.LinkItem = "VSSAVE|Form1"
    Text1.LinkMode = 2
    Text1.LinkExecute "1"
    Text1.LinkMode = 0
End Sub

Sub Form_Load ()
    Rem Nptify Screen Saver was started.
    EnterSaver

    Rem Now we assumes that non-initialized variables have ZERO.
    For i = 0 To 3
        speed(i) = Rnd * 600 + 300 * i
        dire(i) = i
    Next
    speed(1) = -speed(1)
    speed(3) = -speed(3)
End Sub

Sub Timer1_Timer ()
    DoNext
End Sub

Sub DrawIt (x0!, y0!, x1!, y1!, x2!, y2!, x3!, y3!, c&)
    Line (x0!, y0!)-(x1!, y1!), c&
    Line -(x2!, y2!), c&
    Line -(x3!, y3!), c&
    Line -(x0!, y0!), c&
End Sub

Sub DoNext ()

    Rem Erase Oldest Line
    DrawIt xpos(0, 3), ypos(0, 3), xpos(1, 3), ypos(1, 3), xpos(2, 3), ypos(2, 3), xpos(3, 3), ypos(3, 3), col(3)

    Rem Push older positions
    For i = 2 To 0 Step -1
        For j = 0 To 3
            xpos(j, i + 1) = xpos(j, i)
            ypos(j, i + 1) = ypos(j, i)
        Next
        col(i + 1) = col(i)
    Next

    Rem calc new position
    For i = 0 To 3
        Select Case dire(i)
        Case 0
            ypos(i, 0) = 0
            xpos(i, 0) = xpos(i, 0) + speed(i)
            If xpos(i, 0) > form1.width Then dire(i) = 1
            If xpos(i, 0) < 0 Then dire(i) = 3
        Case 1
            xpos(i, 0) = form1.width
            ypos(i, 0) = ypos(i, 0) + speed(i)
            If ypos(i, 0) > form1.height Then dire(i) = 2
            If ypos(i, 0) < 0 Then dire(i) = 0
        Case 2
            ypos(i, 0) = form1.height
            xpos(i, 0) = xpos(i, 0) - speed(i)
            If xpos(i, 0) > form1.width Then dire(i) = 1
            If xpos(i, 0) < 0 Then dire(i) = 3
        Case 3
            xpos(i, 0) = 0
            ypos(i, 0) = ypos(i, 0) - speed(i)
            If ypos(i, 0) > form1.height Then dire(i) = 2
            If ypos(i, 0) < 0 Then dire(i) = 0
        End Select

    Next

    Rem Make New Color, can't be BLACK(0&)
    Do
        If Rnd < .5 Then cl& = &HFF& Else cl& = 0&
        If Rnd < .5 Then cl& = cl& + &HFF00&
        If Rnd < .5 Then cl& = cl& + &HFF0000
        col(0) = cl&
    Loop Until col(0) <> 0&

    Rem Draw New Line
    DrawIt xpos(0, 0), ypos(0, 0), xpos(1, 0), ypos(1, 0), xpos(2, 0), ypos(2, 0), xpos(3, 0), ypos(3, 0), col(0)
End Sub
```

Windows プログラミング用語集
ウィンドウクラス（Window class）　データ構造や操作，機能の同じウィンドウをグループ分けしクラスと呼ぶ．すべてのウィンドウはクラスで分類される．

クリーンセーバー以外の目的にも使用できる．単に，入力がとだえて一定時間経ったときにイベントを発生させるコントロールに過ぎないからだ．

ここでは，こういう発想でカスタムコントロールを開発する方法について述べる．

CDKの入手方法

カスタムコントロールを作成するには，MS-CとSDKとCDKが必要である．

CDKそのものは，秋葉原のラオックスThe Computer館のような場所で買えることもあるが，入手は容易ではない．普通の小売りソフトウェアの体裁を成していないためだ．

それよりも，新しくマイクロソフトから発売されたProfessional Tool Kitにも含まれているので，こちらを入手するほうが楽であろう．こちらは，普通の小売りソフトウェアである．

このキットには，CDKそのものと，CDKで開発されたカスタムコントロールが多数含まれている．また，各種情報の入ったヘルプファイルなども含まれていて，そういう意味でもVBユーザーなら購入する価値がある．

カスタムコントロールの構造

カスタムコントロールは，DLLである．いくつかのVBの要求する手順にそって，VBのAPIを呼び出すように作らなければならない．その手順は，通常のWindows用のDLLとはまったくの別物である．

基本的には，以下のような内容を持つ必要がある．

リスト 22　VSAVER.H

```
//------------------------------------------------------------------------
// Circle.h
//------------------------------------------------------------------------

//------------------------------------------------------------------------
// Helpful Macros
//------------------------------------------------------------------------
#define OFFSETIN(struc,field)     ((USHORT)&(((struc *)0)->field))


//------------------------------------------------------------------------
// Resource ID's
//------------------------------------------------------------------------
// Toolbox bitmap resource IDs numbers.
//------------------------------------------------------------------------
#define IDBMP_CIRCLE          8000
#define IDBMP_CIRCLEDOWN      8001
#define IDBMP_CIRCLEMONO      8003
#define IDBMP_CIRCLEEGA       8006


//------------------------------------------------------------------------
// Constants used by dialog boxes
//------------------------------------------------------------------------
#define DI_OK          1
#define DI_CANCEL      2
#define DI_REDOPT      105
#define DI_GREENOPT    106
#define DI_BLUEOPT     107


//------------------------------------------------------------------------
// Control and Window Procedures
//------------------------------------------------------------------------
LONG FAR PASCAL _export CircleCtlProc(HCTL, HWND, USHORT, USHORT, LONG);
LONG FAR PASCAL _export FlashPopupWndProc(HWND hwnd, USHORT msg, USHORT wp, LONG lp);


//------------------------------------------------------------------------
// Global Variables and Constants
//------------------------------------------------------------------------
extern  HANDLE hmodDLL;      // From ccinit.c
#define CLASS_FLASHPOPUP    "FCPopup"


//------------------------------------------------------------------------
// CIRCLE control data and structs
//------------------------------------------------------------------------
typedef struct tagCIRCLE
{
    RECT    rectDrawInto;
    ENUM    CircleShape;     // Changed from SHORT to demonstrate ENUM prop
    LONG    FlashColor;
    HFONT   hfont;
    HSZ     hszCaption;
    ENUM    bSaving;
    SHORT   WaitSecond;
    SHORT   SampleRate;
    SHORT   TimerID;
} CIRCLE;

typedef CIRCLE FAR * PCIRCLE;


//------------------------------------------------------------------------
// Private messages
//------------------------------------------------------------------------
#define CM_OPENFLASHDLG      WM_USER


#ifndef RC_INVOKED
//------------------------------------------------------------------------
// Property list
//------------------------------------------------------------------------
// Define the consecutive indicies for the properties
//------------------------------------------------------------------------
#define IPROP_CIRCLE_CTLNAME            0
#define IPROP_CIRCLE_INDEX              1
#define IPROP_CIRCLE_BACKCOLOR          2
#define IPROP_CIRCLE_LEFT               3
#define IPROP_CIRCLE_TOP                4
#define IPROP_CIRCLE_WIDTH              5
#define IPROP_CIRCLE_HEIGHT             6
#define IPROP_CIRCLE_VISIBLE            7
#define IPROP_CIRCLE_PARENT             8
#define IPROP_CIRCLE_DRAGMODE           9
#define IPROP_CIRCLE_DRAGICON           10
#define IPROP_CIRCLE_TAG                11
#define IPROP_CIRCLE_CIRCLESHAPE        12
#define IPROP_CIRCLE_FLASHCOLOR         13
#define IPROP_CIRCLE_CAPTION            14
#define IPROP_CIRCLE_FONTNAME           15
#define IPROP_CIRCLE_FONTBOLD           16
#define IPROP_CIRCLE_FONTITALIC         17
```

```
#define IPROP_CIRCLE_FONTSTRIKE          18
#define IPROP_CIRCLE_FONTUNDER           19
#define IPROP_CIRCLE_FONTSIZE            20
#define IPROP_CIRCLE_BORDERSTYLE      21
// bug of original source code
//     PPROPINFO_STD_TABINDEX,        22
//     PPROPINFO_STD_TABSTOP,         23
//     PPROPINFO_STD_ENABLED,         24
//     PPROPINFO_STD_MOUSEPOINTER,    25
#define IPROP_CIRCLE_INSAVING          26
#define IPROP_CIRCLE_WAITSECOND        27
#define IPROP_CIRCLE_SAMPLERATE        28
#define IPROP_CIRCLE_TIMERID           29

// List of enumeration for CircleShape ENUM property
#define SHAPE_CIRCLE      0
#define SHAPE_OVAL    1
#define SHAPE_MAX     1

#ifdef CTL_DATA
PROPINFO Property_CircleShape =
{
    "CircleShape",
    // Changed from SHORT to demonstrate an ENUM property
    DT_ENUM | PF_fGetData | PF_fSetMsg | PF_fSaveData,
    OFFSETIN(CIRCLE, CircleShape), 0,
    SHAPE_CIRCLE,
    "0 - Circle¥0" "1 - Oval¥0", SHAPE_MAX
};

PROPINFO Property_FlashColor =
{
    "FlashColor",
    DT_COLOR | PF_fGetData | PF_fSetData | PF_fSaveData | PF_fEditable,
    OFFSETIN(CIRCLE, FlashColor)
};

PROPINFO Property_InSaving =
{
    "InSaving",
    // Changed from SHORT to demonstrate an ENUM property
    DT_ENUM | PF_fGetData | PF_fSetData | PF_fDefVal,
    OFFSETIN(CIRCLE, bSaving), 0,
    0,
    "0 - Not In Saving¥0" "1 - In Saving¥0", 1
};

PROPINFO Property_WaitSecond =
{
    "WaitSeconds",
    DT_SHORT | PF_fGetData | PF_fSetData | PF_fSaveData | PF_fEditable | PF_fDefVal,
    OFFSETIN(CIRCLE, WaitSecond), 0,
    120
};

PROPINFO Property_SampleRate =
{
    "SampleRate",
    DT_SHORT | PF_fGetData | PF_fSetData | PF_fSetCheck | PF_fSaveData | PF_fEditable | PF_fNo
RuntimeW  | PF_fDefVal,
    OFFSETIN(CIRCLE, SampleRate), 0,
    1
};

PROPINFO Property_TimerID =
{
    "TimerID",
    // Changed from SHORT to demonstrate an ENUM property
    DT_SHORT | PF_fGetData | PF_fSetData | PF_fNoShow | PF_fNoRuntimeW | PF_fDefVal,
    OFFSETIN(CIRCLE, TimerID), 0,
    -1
};

PPROPINFO Circle_Properties[] =
{
    PPROPINFO_STD_CTLNAME,
    PPROPINFO_STD_INDEX,
    PPROPINFO_STD_BACKCOLOR,
    PPROPINFO_STD_LEFT,
    PPROPINFO_STD_TOP,
    PPROPINFO_STD_WIDTH,
    PPROPINFO_STD_HEIGHT,
    PPROPINFO_STD_VISIBLE,
    PPROPINFO_STD_PARENT,
    PPROPINFO_STD_DRAGMODE,
    PPROPINFO_STD_DRAGICON,
    PPROPINFO_STD_TAG,
    &Property_CircleShape,
    &Property_FlashColor,
    PPROPINFO_STD_CAPTION,
    PPROPINFO_STD_FONTNAME,
    PPROPINFO_STD_FONTBOLD,
    PPROPINFO_STD_FONTITALIC,
    PPROPINFO_STD_FONTSTRIKE,
    PPROPINFO_STD_FONTUNDER,
    PPROPINFO_STD_FONTSIZE,
    PPROPINFO_STD_BORDERSTYLEOFF,
    PPROPINFO_STD_TABINDEX,
```

(1) VBに，新しいコントロールを登録する
(2)コントロールのコントロールプロシージャという一種のウィンドウ関数を提供する
(3)プロパティの一覧表を提供する
(4)イベントの一覧表を提供する

これだけ設定しておけば，基本的にVBから，最初から登録済みのコントロールのように扱うことができる．

コントロールプロシージャとは，何かがあったとき，それをカスタムコントロール側に通知する機能である．基本的にウィンドウ関数と同じと考えてよいが，必ずVBからフィルタリングされてから渡され，デフォルト関数に渡す際にはDefWindowProcではなく，VBDefControlProcに渡さなければならない．

プログラムの解説

さて，実際のプログラムの内容を説明する．VBのサンプルプログラムであるcircle3をベースにしているので，これがサポートしている機能はすべて含んでいる．circle3のルーチンについては，CDKのマニュアルに解説があるので(もっとも英語であるが)，ここでは変更点だけ説明する．

まず，vsaver.hである(リスト22)．

ここで，コントロールに関する定義を記述する．これには，プロパティとイベントが含まれる．

まず，プロパティデータを保持する構造体に，必要なメンバを追加する．

たとえば，ここでは，InSaving，WaitSeconds，SampleRate，TimerIDという4つのメンバを追加している．このうち，TimerIDについ

Windows プログラミング用語集
キャレット（Caret）　Windowsではマウスカーソルと区別するために文字カーソルをキャレットと呼ぶ．

ては，VBユーザーからは決して見えないメンバである．

これらのプロパティの属性は，PROPINFO構造体で設定している．

たとえば，TimerIDなら，リスト23のように宣言されている．最初のメンバが，VBユーザーにみえる名前(実際には見せていないが)，2番目はデータ型と属性．DT_SHORTは2バイト整数のデータ型である．そして，PF_fGetDataとPF_fSetDataは，値を直接調べてよく，値が変更されたらそのまま値を構造体に格納してよいという指定である．これは，値の読みだした設定が何かのアクションのトリガになるようなプロパティの場合は，メッセージを送らせるように設定できる．PF_fNpoShowはVBユーザーに見えないようにする指定．PF_fNoRuntimeは，実行中の変更不許可である．最後のPF_fDefValはデフォルトの値を持つという指定である．

3番目のメンバは，構造体の中でのオフセット位置である．4番目は，1つのメンバに複数のプロパティ値をパックするときの指定（ここでは使っていない)．5番目はデフォルト値そのものである．

さて，次にイベントであるが，StartSaverのイベントは引数もなく簡単である（リスト24)．

これをイベントテーブルに入れておけば，しっかりとイベントとして認知され，コード入力ウィンドウなどから，このイベントに対応したルーチンが書き込めるようになる．

さて，次はccinit.cである．これは，初期化のための定義を集めたソースファイルであるが，ここでは重要なコードを入れている．

ずばり，システムキューをフックしてメッセージを横取りする処理である．

```
    PPROPINFO_STD_TABSTOP,
    PPROPINFO_STD_ENABLED,
    PPROPINFO_STD_MOUSEPOINTER,
    &Property_InSaving,
    &Property_WaitSecond,
    &Property_SampleRate,
    &Property_TimerID,
    NULL
};
#endif  // CTL_DATA


//-----------------------------------------------------------------------
// Event list
//-----------------------------------------------------------------------
// Define the consecutive indicies for the events
//-----------------------------------------------------------------------
#define IEVENT_CIRCLE_START       0
#define IEVENT_CIRCLE_CLICKIN         1
#define IEVENT_CIRCLE_CLICKOUT        2
#define IEVENT_CIRCLE_DRAGDROP        3
#define IEVENT_CIRCLE_DRAGOVER        4


#ifdef CTL_DATA
WORD Paramtypes_ClickIn[] = {ET_R4, ET_R4, ET_SD};

EVENTINFO Event_ClickIn =
{
    "ClickIn",
    3,
    6,
    Paramtypes_ClickIn,
    "X As Single,Y As Single,Caption As String"
};

EVENTINFO Event_ClickOut =
{
    "ClickOut",
    0,
    0,
    NULL,
    NULL
};

EVENTINFO Event_Start =
{
    "StartSaver",
    0,
    0,
    NULL,
    NULL
};

PEVENTINFO Circle_Events[] =
{
    &Event_Start,
    &Event_ClickIn,
    &Event_ClickOut,
    PEVENTINFO_STD_DRAGDROP,
    PEVENTINFO_STD_DRAGOVER,
    NULL
};
#endif  // CTL_DATA


//-----------------------------------------------------------------------
// Model struct
//-----------------------------------------------------------------------
// Define the control model (using the event and property structures).
//-----------------------------------------------------------------------
#ifdef CTL_DATA
MODEL modelCircle =
{
    VB_VERSION,                 // VB version being used
    0,                      // MODEL flags
    (PCTLPROC)CircleCtlProc,          // Control procedure
    CS_VREDRAW | CS_HREDRAW,          // Class style
    0L,                     // Default Windows style
    sizeof(CIRCLE),             // Size of CIRCLE structure
    IDBMP_CIRCLE,               // Palette bitmap ID
    "ScreenSaver",                  // Default control name
    "VSave1",                   // Visual Basic class name
    NULL,                   // Parent class name
    Circle_Properties,              // Property information table
    Circle_Events               // Event information table
};
#endif  // CTL_DATA

extern DWORD dwLastAccessTime;

#endif  // RC_INVOKED

//-----------------------------------------------------------------------
```

VBとは直接関係はないが説明しておく．

まず，このDLLが初期化された時点で，フックを掛けなければならない．そして，DLLがなくなるときには，フックを戻さなければならない．その処理と，LibMainとWEPに入れてみたのだが，うまく動作しなかった．そのため，VBINITCCとVBTERMCCに入れてみた．その場合，問題なのは，2回以上呼び出される可能性があることだ．そのため，何回呼び出されたかをカウントする変数を持ち，一番最初のVBINITCCと一番最後のVBTERMCCのときだけ実行させるようにした．

実際のフックは，リスト25の1行で行っている．DLLの場合は，MakeProcInstanceは不要である．また，WH_JOURNALRECORDとは，もともと，レコーダーのような，マウスやキーボードのイベントを保管して再現するためのツール用のフックである．そのため，マウスとキーボードで起こったことは，すべて通知してくれるという，まさにスクリーンセーバー向きのフックである．

フックの解除は，リスト26のように行う．

さて，このフックによってフックさせたルーチンは，リスト27のようにやって，何かが起こった時刻（TICK単位）を記録しておく．この時刻と現在時刻の差がWaitSeconds＊1000よりも大きくなれば，StartSaverイベントを起こすことになる．

さて，最後はvsaver.cである（リスト28）．

全部説明すると長くなるので，要点だけかいつまんで説明する．

基本的に，プロパティへの値の設定と読みだしは，VBのAPIである，VBSetControlPropertyとVBGetControlPropertyによって行っている．

また，WM_CREATEとWM_DESTROYで，タイマのスタートと停止を行っている．タイマはコールバック関数ではなく，WM_TIMERで渡すようにしている．これは，スクリーンセーバーの処理は遅くなっても構わないためである．

タイマメッセージの処理ルーチンの中で，イベントの発生する可能性のある2つの条件を調べている．

1つは，最後の入力からの経過時間である．これは，リスト29で調べている．

また，もう1つの条件，マウスポインタが画面の右上隅という条件も調べている．ただ，ずっとマウスポインタが右上隅にいると連続してイベントが走ってしまったりする可能性があるので，右上隅ではない座標から右上隅に移動した瞬間のみをつかまえてイベントを発生させている．

イベントそのものの発生は，リスト30のようにVBのAPIであるVBFireEventの呼び出しで行っている．

最後に，細かいことであるが，サンプリングレートが変更されたときは，一度タイマを止めて新しいレートで再

リスト 23

```
PROPINFO Property_TimerID =
{
    "TimerID",
    DT_SHORT | PF_fGetData | PF_fSetData | PF_fNoShow | PF_fNoRuntimeW | PF_fDefVal,
    OFFSETIN(CIRCLE, TimerID), 0,
    -1
};
```

リスト 24

```
EVENTINFO Event_Start =
{
    "StartSaver",
    0,
    0,
    NULL,
    NULL
};
```

リスト 25

```
OldHookProc = SetWindowsHook(
        WH_JOURNALRECORD,
        (FARPROC)HookProc
);
```

リスト 26

```
UnhookWindowsHook(
        WH_JOURNALRECORD,
        (FARPROC)HookProc
);
```

リスト 27

```
LPEVENTMSGMSG lpMsg;
lpMsg = (LPEVENTMSGMSG)lParam;
dwLastAccessTime = lpMsg->time;
```


Windows プログラミング用語集
キュー（Queues）　Windowsで利用されるおもなメッセージはいったんキューと呼ばれる一種のバッファに格納され，プログラムやシステムに送られる．

リスト 28 VSAVER.C

```
//-----------------------------------------------------------------
//      Copyright (C) 1991 Microsoft Corporation
//
// You have a royalty-free right to use, modify, reproduce and distribute
// the Sample Custom Control Files (and/or any modified version) in any way
// you find useful, provided that you agree that Microsoft has no warranty,
// obligation or liability for any Custom Control File.
//-----------------------------------------------------------------
// Circle.c
//-----------------------------------------------------------------
// Contains control procedure for CIRCLE3 control
//-----------------------------------------------------------------

#define  NOCOMM
#include <windows.h>

#include <vbapi.h>
#include <string.h>
#include "vsaver.h"

#include <assert.h>

//-----------------------------------------------------------------
// Local Prototypes
//-----------------------------------------------------------------
VOID NEAR PaintCircle(PCIRCLE pcircle, HWND hwnd, HDC hdc);
VOID NEAR RecalcArea(PCIRCLE pcircle, HWND hwnd);
VOID NEAR FlashCircle(PCIRCLE pcircle, HDC hdc);
BOOL NEAR InCircle(PCIRCLE pcircle, SHORT xcoord, SHORT ycoord);
VOID NEAR FireClickIn(HCTL hctl, HWND hwnd, SHORT x, SHORT y);
VOID NEAR FireClickOut(HCTL hctl);
VOID NEAR FireStartSaver(HCTL hctl);
HWND NEAR HwndInitFlashPopup(VOID);
BOOL _export FAR PASCAL FlashDlgProc(HWND hDlg, USHORT msg, USHORT wp, LONG lp);

//-----------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------
WORD ipropDialog;
HCTL hctlDialog;

static VOID NEAR PASCAL StartInterval(
    HCTL    hctl,
    HWND    hwnd,
    SHORT   sr
)
{
    WORD wTimerID;

    wTimerID = SetTimer( hwnd, 1, sr*1000, NULL );
    if( wTimerID == 0 ) {
        MessageBox(
            NULL,
            "OUT OF TIMER! CAN'T USE SCREEN SAVER",
            "VSAVER",
            MB_OK|MB_ICONEXCLAMATION
        );
    }
    VBSetControlProperty(
        hctl,
        IPROP_CIRCLE_TIMERID,
        wTimerID
    );

}

static VOID NEAR PASCAL StartIntervalSR(
    HCTL    hctl,
    HWND    hwnd
)
{
    SHORT sr;

    VBGetControlProperty(
        hctl,
        IPROP_CIRCLE_SAMPLERATE,
        &sr
    );

    // don't set value less than 1sec.
    sr = max( 1, sr );

    StartInterval( hctl, hwnd, sr );
}

static VOID NEAR PASCAL StopInterval(
    HCTL    hctl
)
{
    WORD wTimerID;

    VBGetControlProperty(
        hctl,
        IPROP_CIRCLE_TIMERID,
        &wTimerID
    );
    if( wTimerID != 0 ) KillTimer( wTimerID, NULL );

    wTimerID = 0;
    VBSetControlProperty(
        hctl,
        IPROP_CIRCLE_TIMERID,
        wTimerID
    );
}

static POINT oldPnt = {0,0};

static VOID NEAR PASCAL WmTimer(
    HCTL    hctl,
    HWND    hwnd
)
{
    SHORT ws;
    SHORT InSaving;

    VBGetControlProperty(
        hctl,
        IPROP_CIRCLE_WAITSECOND,
        &ws
    );
    if( GetTickCount() - dwLastAccessTime  > (DWORD)ws*1000L ) {
        VBGetControlProperty(
            hctl,
            IPROP_CIRCLE_INSAVING,
            &InSaving
        );
        if( InSaving == 0 ) {
            StopInterval( hctl );
            FireStartSaver( hctl );
            dwLastAccessTime = GetTickCount();
            StartIntervalSR( hctl, hwnd );
        } else {
            dwLastAccessTime = GetTickCount();
        }
    } else {
        POINT pnt;
        GetCursorPos( &pnt );
        if( pnt.x == GetSystemMetrics( SM_CXSCREEN ) - 1
         && pnt.y == 0
         && (oldPnt.x != GetSystemMetrics( SM_CXSCREEN ) - 1
           || oldPnt.y != 0 ) ) {
            if( InSaving == 0 ) {
                StopInterval( hctl );
                FireStartSaver( hctl );
                dwLastAccessTime = GetTickCount();
                StartIntervalSR( hctl, hwnd );
            } else {
                dwLastAccessTime = GetTickCount();
            }
        }
        oldPnt = pnt;
    }
}

//-----------------------------------------------------------------
// Circle Control Procedure
//-----------------------------------------------------------------
LONG FAR PASCAL _export CircleCtlProc
(
    HCTL   hctl,
    HWND   hwnd,
    USHORT msg,
    USHORT wp,
    LONG   lp
)
{
    PCIRCLE pcircle = NULL;
    HDC     hdc;

    switch (msg)
    {
        case WM_CREATE:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        pcircle->CircleShape = SHAPE_CIRCLE;
            pcircle->FlashColor = 0L;
        VBSetControlProperty(hctl, IPROP_CIRCLE_BACKCOLOR, 255L);
        // *** pcircle may now be invalid due to call to VB API ***

        dwLastAccessTime = GetTickCount();
        StartIntervalSR( hctl, hwnd );

            break;

    case WM_DESTROY:
        StopInterval( hctl );
        break;

    case WM_TIMER:
        WmTimer( hctl, hwnd );
        break;

        case WM_LBUTTONDOWN:
        case WM_LBUTTONDBLCLK:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        if (InCircle(pcircle, LOWORD(lp), HIWORD(lp)))
        {
        hdc = GetDC(hwnd);
        FlashCircle(pcircle, hdc);
        ReleaseDC(hwnd, hdc);
        FireClickIn(hctl, hwnd, LOWORD(lp), HIWORD(lp));
        // *** pcircle may now be invalid due to call to VB API ***
        // *** inside this function                             ***
        }
        else
        FireClickOut(hctl);
```

```
        // *** pcircle may now be invalid due to call to VB API ***
        // *** inside this function                          ***
            break;

        case WM_LBUTTONUP:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        if (InCircle(pcircle, LOWORD(lp), HIWORD(lp)))
        {
        hdc = GetDC(hwnd);
        PaintCircle(pcircle, hwnd, hdc);
        ReleaseDC(hwnd, hdc);
        }
            break;

        case WM_SETFONT:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        pcircle->hfont = (HFONT)wp;
            return 0;

        case WM_GETFONT:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        return pcircle->hfont;

        case WM_SETTEXT:
            {
        HSZ hsz;

        pcircle = (PCIRCLE)VBDerefControl(hctl);
        if (pcircle->hszCaption)
        {
        VBDestroyHsz(pcircle->hszCaption);
        // *** pcircle may now be invalid due to call to VB API ***
        }
        hsz = VBCreateHsz((_segment)hctl, (LPSTR)lp);
        // *** pcircle may now be invalid due to call to VB API ***
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        pcircle->hszCaption = hsz;
        InvalidateRect(hwnd, NULL, TRUE);
        return 0L;
            }

        case WM_GETTEXT:
            {
        LPSTR   lpstr;
        WORD    cch;

        pcircle = (PCIRCLE)VBDerefControl(hctl);
        if (pcircle->hszCaption == NULL)
        {
        *(LPSTR)lp = 0L;
        wp = 1;
        }
        else
        {
        lpstr = VBDerefHsz(pcircle->hszCaption);
        cch = lstrlen(lpstr) + 1;
        if (wp > cch)
            wp = cch;
        _fstrncpy((LPSTR)lp, lpstr, wp);
        ((LPSTR)lp)[wp - 1] = '¥0';
        }
            }
        return (LONG)(wp - 1);

        case WM_GETTEXTLENGTH:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
            if (pcircle->hszCaption == NULL)
                return 0L;
            else
        return lstrlen(VBDerefHsz(pcircle->hszCaption));


        case WM_PAINT:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
            if (wp)
        PaintCircle(pcircle, hwnd, (HDC)wp);
        else
        {
                PAINTSTRUCT ps;

        BeginPaint(hwnd, &ps);
        PaintCircle(pcircle, hwnd, ps.hdc);
        EndPaint(hwnd, &ps);
        }
            break;

        case WM_SIZE:
        pcircle = (PCIRCLE)VBDerefControl(hctl);
        RecalcArea(pcircle, hwnd);
            break;

        case VBM_SETPROPERTY:
        switch (wp)
        {
                case IPROP_CIRCLE_CIRCLESHAPE:
            pcircle = (PCIRCLE)VBDerefControl(hctl);
            pcircle->CircleShape = (ENUM)lp;
            RecalcArea(pcircle, hwnd);
            InvalidateRect(hwnd, NULL, TRUE);
            return 0L;
        }
            break;

    case VBM_CHECKPROPERTY:
        switch( wp ) {
            case IPROP_CIRCLE_SAMPLERATE:
            StopInterval( hctl );
            StartInterval( hctl, hwnd, (SHORT)lp );
            return 0L;
        }
            break;

        case VBM_INITPROPPOPUP:
        switch (wp)
        {
        // Un-commenting the following line will enable our custom
        // popup instead of the color palette, when setting the
        // backcolor:
        // case IPROP_CIRCLE_BACKCOLOR:
        case IPROP_CIRCLE_FLASHCOLOR:
            hctlDialog  = hctl;
            ipropDialog = wp;
            return (LONG)HwndInitFlashPopup();
                }
            break;
    }

    return VBDefControlProc(hctl, hwnd, msg, wp, lp);
}


//---------------------------------------------------------------------------
// Handle painting by drawing circle into the given hdc.
//---------------------------------------------------------------------------
VOID NEAR PaintCircle
(
    PCIRCLE pcircle,
    HWND    hwnd,
    HDC     hdc
)
{
    HBRUSH  hbr;
    HBRUSH  hbrOld = NULL;
    LPSTR   lpstr;
    LPRECT  prect = &pcircle->rectDrawInto;
    HFONT   hfontOld;

    hbr = (HBRUSH)SendMessage(GetParent(hwnd), WM_CTLCOLOR,
                hdc, MAKELONG(hwnd, 0));
    hbrOld = SelectObject(hdc, hbr);
    Ellipse(hdc, prect->left, prect->top, prect->right, prect->bottom);

    hfontOld = SelectObject(hdc, pcircle->hfont);
    lpstr = VBDerefHsz(pcircle->hszCaption);
    DrawText(hdc, lpstr, -1, prect, DT_VCENTER | DT_CENTER | DT_SINGLELINE);

    SelectObject(hdc, hfontOld);
    SelectObject(hdc, hbrOld);
}


//---------------------------------------------------------------------------
// Paint the circle in the FlashColor.
//---------------------------------------------------------------------------
VOID NEAR FlashCircle
(
    PCIRCLE pcircle,
    HDC     hdc
)
{
    HBRUSH  hbr;
    HBRUSH  hbrOld = NULL;
    LPRECT  prect = &pcircle->rectDrawInto;

    hbr = CreateSolidBrush(RGBCOLOR(pcircle->FlashColor));
    if (hbr)
    hbrOld = SelectObject(hdc, hbr);
    Ellipse(hdc, prect->left, prect->top, prect->right, prect->bottom);
    if (hbr)
    {
    SelectObject(hdc, hbrOld);
    DeleteObject(hbr);
    }
}


//---------------------------------------------------------------------------
// Use the hwnd's client size to determine the bounding rectangle for the
// circle.  If CircleShape is TRUE, then we need to calculate a square
// centered in prect.
//---------------------------------------------------------------------------
VOID NEAR RecalcArea
(
    PCIRCLE pcircle,
    HWND    hwnd
)
{
    LPRECT  prect = &pcircle->rectDrawInto;

    GetClientRect(hwnd, prect);
    if (pcircle->CircleShape == SHAPE_OVAL)
        return;
    if (prect->right > prect->bottom)
    {
        prect->left = (prect->right - prect->bottom) / 2;
        prect->right = prect->left + prect->bottom;
    }
    else if (prect->bottom > prect->right)
    {
        prect->top = (prect->bottom - prect->right) / 2;
        prect->bottom = prect->top + prect->right;
    }
}


//---------------------------------------------------------------------------
// Return TRUE if the given coordinates are inside of the circle.
//---------------------------------------------------------------------------
BOOL NEAR InCircle
(
```


Windows プログラミング用語集
クラススコープ（Class scope）　データ構造とそれらに作用する関数も含めた特定のメンバの集まりであるクラスの適用範囲を指す。

```
    PCIRCLE pcircle,
    SHORT   xcoord,
    SHORT   ycoord
)
{
    double  a, b;
    double  x, y;
    LPRECT  prect = &pcircle->rectDrawInto;

    a = (prect->right - prect->left) / 2;
    b = (prect->bottom - prect->top) / 2;
    x = xcoord - (prect->left + prect->right) / 2;
    y = ycoord - (prect->top + prect->bottom) / 2;
    return ((x * x) / (a * a) + (y * y) / (b * b) <= 1);
}


//---------------------------------------------------------------------------
// TYPEDEF for parameters to the ClickIn event.
//---------------------------------------------------------------------------
typedef struct tagCLICKINPARMS
{
    HLSTR       ClickString;
    float far *Y;
    float far *X;
    LPVOID      Index;
} CLICKINPARMS;


//---------------------------------------------------------------------------
// Fire the ClickIn event, passing the x,y coords of the click.  Also pass
// the current caption of the Circle control, to demonstrate passing strings
// to event procedures.
//---------------------------------------------------------------------------
VOID NEAR FireClickIn
(
    HCTL  hctl,
    HWND  hwnd,
    SHORT x,
    SHORT y
)
{
    CLICKINPARMS params;
    float    VBx, VBy;
    int      cbCaption, err;
    char     strBuf[20];

    VBx = (float)VBXPixelsToTwips(x);
    VBy = (float)VBYPixelsToTwips(y);
    params.X = &VBx;
    params.Y = &VBy;

    cbCaption = GetWindowText(hwnd, strBuf, 20);
    params.ClickString = VBCreateHlstr(strBuf, cbCaption);
    err = VBFireEvent(hctl, IEVENT_CIRCLE_CLICKIN, &params);
    VBDestroyHlstr(params.ClickString);
}


//---------------------------------------------------------------------------
// Fire the Saver event.
//---------------------------------------------------------------------------
VOID NEAR FireStartSaver
(
    HCTL hctl
)
{
    VBFireEvent(hctl, IEVENT_CIRCLE_START, NULL);
}


//---------------------------------------------------------------------------
// Fire the ClickOut event.
//---------------------------------------------------------------------------
VOID NEAR FireClickOut
(
    HCTL hctl
)
{
    VBFireEvent(hctl, IEVENT_CIRCLE_CLICKOUT, NULL);
}


//---------------------------------------------------------------------------
// Create our property popup-window.  Since we want to put up a dialog, this
// window never becomes visible.  Instead, when asked to become visible, it
// will post a message to itself, remining it to put up our dialog.
//
// NOTE: May return NULL!
//---------------------------------------------------------------------------
HWND NEAR HwndInitFlashPopup
(
    VOID
)
{
    return CreateWindow(CLASS_FLASHPOPUP, NULL, WS_POPUP,
                0, 0, 0, 0, NULL, NULL,
                hmodDLL, NULL);
}


//---------------------------------------------------------------------------
// We asked to show ourself, remain invisible and post a CM_OPENFLASHDLG to
// ourself.  When we receive this message, open the dialog box.
//---------------------------------------------------------------------------
LONG _export FAR PASCAL FlashPopupWndProc
(
    HWND   hwnd,
    USHORT msg,
    USHORT wp,
    LONG   lp
)
{
    extern HANDLE hmodDLL;

    switch (msg)
    {
        case WM_SHOWWINDOW:
        if (wp)
        {
        PostMessage(hwnd, CM_OPENFLASHDLG, 0, 0L);
        return 0L;
        }
            break;

    case CM_OPENFLASHDLG:
        VBDialogBoxParam(hmodDLL, "FlashDlg", FlashDlgProc, 0L);
        return 0L;
    }

    return DefWindowProc(hwnd, msg, wp, lp);
}


//---------------------------------------------------------------------------
// An array mapping option buttons to RGB colors.
//---------------------------------------------------------------------------
long mpidcolor[] = { 0xff, 0xff00, 0xff0000 };


//---------------------------------------------------------------------------
// The Dialog Procedure for the FlashColor property dialog.
//---------------------------------------------------------------------------
BOOL FAR PASCAL _export FlashDlgProc
(
    HWND   hDlg,
    USHORT msg,
    USHORT wp,
    LONG   lp
)
{
    static LONG colorOld;

    switch (msg)
    {
    case WM_INITDIALOG:
        {
        RECT rect;
        int  nx, ny;      // New x and y
        int  width, height;
        int  i;

        // Position dialog so it looks nice:
        GetWindowRect(hDlg, &rect);
        width  = rect.right - rect.left;
        height = rect.bottom - rect.top;
        nx = (GetSystemMetrics(SM_CXSCREEN) - width)  / 2;
        ny = (GetSystemMetrics(SM_CYSCREEN) - height) / 3;
        MoveWindow(hDlg, nx, ny, width, height, FALSE);

        // Remember the old value of this property, so we can restore it
        // on cancel:
        if (VBGetControlProperty(hctlDialog, ipropDialog, &colorOld))
          EndDialog(hDlg, FALSE);

        // If the current color matches one of the option button colors,
        // then set that option button:
        for (i=0; i<sizeof(mpidcolor); i++)
        if (mpidcolor[i] == colorOld)
            CheckRadioButton(hDlg, DI_REDOPT, DI_BLUEOPT, i+DI_REDOPT);

        return TRUE;
        }

        case WM_COMMAND:
        switch (wp)
        {
                case IDOK:
            EndDialog(hDlg, TRUE);
                    return TRUE;

                case IDCANCEL:
            EndDialog(hDlg, FALSE);
            // Restore the old value, since we're canceling:
            VBSetControlProperty(hctlDialog, ipropDialog, colorOld);
                    return TRUE;

                case DI_REDOPT:
                case DI_GREENOPT:
                case DI_BLUEOPT:
            CheckRadioButton(hDlg, DI_REDOPT, DI_BLUEOPT, wp);
            VBSetControlProperty(hctlDialog, ipropDialog,
                     mpidcolor[wp-DI_REDOPT]);
                    return TRUE;
        }
            break;
    }
    return FALSE;
}

//---------------------------------------------------------------------------
```

Windows プログラミング用語集

クリップボード（Clip board）　プログラム間のデータ転送など，メモリブロックの交換を行う Windows関数，同名のWindowsアプリケーションもある。

スタートさせなければならない．

まず，SampleRateプロパティにPF_fSetCheck属性を付けておく．そうすると，SampleRateに値が代入されるたびにVBM_CHECKPROPERTYメッセージが送られてくる．その時点で，タイマの停止と再スタートを行えばだいじょうぶである．

最後に忘れてならないのが，defファイルの修正である．vsaver.defのモジュール名を，きちんと，VSAVER1に直しておく．

さて，最後に，makefileを見てみよう（リスト31）．

最初に，2つのマクロが定義されている．

```
PRODUCT = VSAVER
VERSION = 1
```

PRODUCTは，基本的にこれから作りたいファイルの名前である．それに，VERSIONの値を結合して使っている．なぜ，VERSIONなどという数字を結合しているのだろうか．

Windowsでは，同一のモジュール名のモジュールは，EXEファイルであれDLLであれ，2回ロードされることはない．したがって，同じ名前なのに違う内容を持ったモジュールが存在すると非常に困る．もし，カスタムコントロールを改良して，互換性がなくなってしまったとしよう．すると，旧バージョンのカスタムコントロールで作ったアプリと，新バージョンのカスタムコントロールで作ったアプリを同時に起動できないことになる．さす

リスト 29

```
if( GetTickCount() - dwLastAccessTime  > (DWORD)ws*1000L ) {
```

リスト 30

```
VOID NEAR FireStartSaver
(
    HCTL hctl
)
{
    VBFireEvent(hctl, IEVENT_CIRCLE_START, NULL);
}
```

リスト 31 MAKEFILE

```
#
# Custom Control makefile
#

.SUFFIXES:  .asm .c .def .VBX .h .lnk .map .obj .rc .res .sym

PRODUCT = VSAVER
VERSION = 1

Default: $(PRODUCT)$(VERSION).VBX

#
# Tools required
#

AS          = masm
CC          = cl
CVPACK      = cvpack
LINK        = link
MAPSYM      = mapsym
RC          = $(RCPATH)rc
#SZSCRN     = echo
SZSCRN      = echo >con

#
# Flags
#

AFLAGS      = -W2 -V -E -P -Zi
CFLAGS      = -W3 -c -G2csw -Alnw -Zip
LFLAGS      = /co /align:16 /batch /far /li /map /nod /noe /nop /w

#
# DEFAULT BUILD RULES
#

.c.obj:
    $(SZSCRN) Compiling $(<F)
    if exist $@ del $@
    $(CC) $(CFLAGS) $<

.asm.obj:
    $(SZSCRN) Assembling $(<F)
    $(AS) $(AFLAGS) $<,$@;

#
# Dependencies
#

ccinit.obj:     ccinit.c $(PRODUCT).h

libinit.obj:    libinit.asm

$(PRODUCT).obj: $(PRODUCT).c $(PRODUCT).h

$(PRODUCT)$(VERSION).VBX: libinit.obj ccinit.obj $(PRODUCT).obj $(PRODUCT).lnk ¥
        $(PRODUCT).res $(PRODUCT).def
    $(SZSCRN) Linking $(PRODUCT)$(VERSION).VBX...
    $(LINK) /co @$(PRODUCT).lnk
    $(SZSCRN) CVPACKing $(PRODUCT)$(VERSION).VBX...
    $(CVPACK) -p $(PRODUCT)$(VERSION).VBX
    $(SZSCRN) RCing $(PRODUCT)$(VERSION).VBX...
    $(RC) $(PRODUCT).res $(PRODUCT)$(VERSION).VBX
    $(SZSCRN) MAPSYMing $(PRODUCT)$(VERSION).VBX...
    $(MAPSYM) $(PRODUCT)$(VERSION)
    $(SZSCRN) Done Linking $(PRODUCT)$(VERSION).VBX

$(PRODUCT).lnk: makefile
    $(SZSCRN) Making <<$(PRODUCT).lnk
    libinit.obj+
    ccinit.obj+
    $(PRODUCT).obj
    $(PRODUCT)$(VERSION).VBX $(LFLAGS)
    $(PRODUCT)$(VERSION).map
    vbapi.lib mdllcew.lib libw.lib
    $(PRODUCT).def
<<KEEP

$(PRODUCT).res: $(PRODUCT).rc $(PRODUCT).h ¥
        circlecd.bmp ¥
        circlecu.bmp ¥
        circlemu.bmp ¥
        circleeu.bmp
    $(SZSCRN) Resource compiling $(PRODUCT).RC
    $(RC) -R -i$(TOOLS)¥inc $(RCINCS) $(PRODUCT).rc

cln:
    -del *.obj
    -del *.res
    -del *.lnk
    -del *.vbx
    -del *.map
    -del *.sym
```


Windows プログラミング用語集
グラフィックデバイスインターフェース（Graphic device interface（GDI））　デバイス・プログラム間で，プログラムのビジュアルな情報処理を支援する．

がに，これは実用的ではない．したがって，仕様が変わったらファイル名(モジュール名)も変える．これが安全策である．同じようなことが，VB自身のランタイムライブラリDLLにもいえる．VBRUN100.DLLというファイル名は，バージョンアップしたらファイル名を変更することが容易に予測できる．

最後に

さて，かなりおおざっぱになってしまったが，VBで実際にアプリケーションを開発する手順について述べてみた．VBは，CDKを使うことによって無限の可能性を見せてくれることがわかった．このことから，今後数年で，何か凄いことが起こるのではないかと思うのだが，何が起こるのか予測もできない．もしかしたら，その凄いことを起こすのは，これを読んでいるあなたかもしれない．

参考文献

(1)「Windows工房の仲間たち　Visual-Basicで遊ぶ日曜日」，きたあきら，ソフトバンク THE WINDOWS 1992年5月号

リスト 32　LIBINIT.ASM

```
title   LibInit.asm - Windows DynLink Library Initialization

;--------------------------------------------------------------------

extrn   LibMain:far

extrn   __acrtused:ABS  ;make sure C-runtime gets pulled in

;--------------------------------------------------------------------

LIBINIT_TEXT  SEGMENT BYTE PUBLIC 'CODE'
ASSUME  CS:LIBINIT_TEXT

;--------------------------------------------------------------------
; DLL entry point.
;   This routine is called when the first client loads the DLL.
;--------------------------------------------------------------------
PUBLIC      astartLib
astartLib   PROC FAR
    push    di      ;hmodLib
    push    ds      ;segDS
    push    cx      ;usHeapSize
    call    LibMain
    ret
astartLib   ENDP

LIBINIT_TEXT ENDS

;--------------------------------------------------------------------

        end     astartLib
```

リスト 33　VSAVER.BAS

```
Declare Sub WriteProfileString Lib "kernel" (ByVal appname As String, ByVal keyname As String, ByVal value As String)
Declare Function GetProfileString Lib "kernel" (ByVal Sname$, ByVal Kname$, ByVal def$, ByVal ret$, ByVal size%) As Integer
Declare Sub ShowCursor Lib "user" (ByVal bShow As Integer)

```

リスト 34　CCINIT.C

```
//--------------------------------------------------------------------
// ccInit.c
//--------------------------------------------------------------------
// Custom Control DLL Initialization
//--------------------------------------------------------------------

#define  NOCOMM
#include <windows.h>

#include <vbapi.h>

#define  CTL_DATA       // To declare static data in circle.h
#include "vsaver.h"


//--------------------------------------------------------------------
// Global Variables
//--------------------------------------------------------------------
HANDLE hmodDLL;
HANDLE taskDevEnvironment = NULL;


// global defenitions, by A.K.
FARPROC MyHookProc = NULL;
FARPROC OldHookProc = NULL;
DWORD dwLastAccessTime = 0;
int ReferenceCount = 0;

// hook filter function, by A.K.
DWORD FAR PASCAL _export HookProc( int nCode, WORD wParam, DWORD lParam )
{
    LPEVENTMSGMSG lpMsg;

    if( nCode != HC_ACTION ) {
        return( DefHookProc( nCode, wParam, lParam, &OldHookProc ) );
    }

    lpMsg = (LPEVENTMSGMSG)lParam;
    dwLastAccessTime = lpMsg->time;

    return( 0L );
}


//--------------------------------------------------------------------
// Register custom control.
//  This routine is called by VB when the custom control DLL is
//  loaded for use.
// Also, register the property popup used to set FlashColor, if this load
//  is from the development environment.
//--------------------------------------------------------------------
BOOL FAR PASCAL _export VBINITCC
(
    USHORT usVersion,
    BOOL   fRuntime
)
{
    if( ReferenceCount == 0 ) {
        OldHookProc = SetWindowsHook(
            WH_JOURNALRECORD,
```

Windows プログラミング用語集

グローバルメモリ（Global memory）　MS-DOSがWindowsをロードした先頭から物理メモリの最終メモリのアドレスを指す．

```
                (FARPROC)HookProc
            );
        }
        ReferenceCount ++;

        // Avoid warnings on unused (but required) formal parameters
        usVersion = usVersion;

        // Register popup class if this is from the development environment.
        if (!fRuntime)
        {
        WNDCLASS class;

        class.style        = 0;
        class.lpfnWndProc   = FlashPopupWndProc;
        class.cbClsExtra    = 0;
        class.cbWndExtra    = 0;
        class.hInstance     = hmodDLL;
        class.hIcon        = NULL;
        class.hCursor        = NULL;
            class.hbrBackground = NULL;
        class.lpszMenuName  = NULL;
        class.lpszClassName = CLASS_FLASHPOPUP;

        if (!RegisterClass(&class))
            return FALSE;

        // Remember the task associated with the development environment
        taskDevEnvironment = GetCurrentTask();
        }

        // Register control(s)
        return VBRegisterModel(hmodDLL, &modelCircle);
}

//---------------------------------------------------------------------------
// Unregister the property popup used to set FlashColor, if this unload
//  is from the development environment.
//---------------------------------------------------------------------------
VOID FAR PASCAL _export VBTERMCC
(
)
{
        ReferenceCount --;
        if( ReferenceCount == 0 ) {
            UnhookWindowsHook( WH_JOURNALRECORD, (FARPROC)HookProc );
        }

        // Unregister popup class if this is from the development environment
        if (taskDevEnvironment == GetCurrentTask())
        {
        UnregisterClass(CLASS_FLASHPOPUP, hmodDLL);
        taskDevEnvironment = NULL;
        }
}

//---------------------------------------------------------------------------
// Initialize library.
//  This routine is called from the DLL entry point in LIBINIT.ASM
//  which is called when the first client loads the DLL.
//---------------------------------------------------------------------------
BOOL FAR PASCAL LibMain
(
    HANDLE hmod,
    HANDLE segDS,
    USHORT cbHeapSize
)
{
    // Avoid warnings on unused (but required) formal parameters
    cbHeapSize = cbHeapSize;
    segDS = segDS;

    hmodDLL = hmod;

    // Leave our DS unlocked when we're not running
    UnlockData( 0 );

    return TRUE;
}


//---------------------------------------------------------------------------
// Handle exit notification from Windows.
//  This routine is called by Windows when the library is freed
//  by its last client.
//---------------------------------------------------------------------------
VOID FAR PASCAL _export WEP
(
    BOOL fSystemExit
)
{
    // Avoid warnings on unused (but required) formal parameters
    fSystemExit = fSystemExit;
}

//---------------------------------------------------------------------------
```

リスト 35 VSAVER.DEF

```
libinit.obj+
ccinit.obj+
VSAVER.obj
VSAVER1.VBX /co /align:16 /batch /far /li /map /nod /noe /nop /w
VSAVER1.map
vbapi.lib mdllcew.lib libw.lib
VSAVER.def
```

リスト 36 VSAVER.LNK

```
;---------------------------------------------------------------------------
; Circle.def - module definition file for CIRCLE3.VBX custom control
;---------------------------------------------------------------------------

LIBRARY      VSAVER1
EXETYPE        WINDOWS
DESCRIPTION 'Visual Basic Circle Custom Control'

CODE         MOVEABLE
DATA         MOVEABLE SINGLE

HEAPSIZE     1024

EXPORTS      FlashDlgProc

;---------------------------------------------------------------------------
```

リスト 37 VSAVER.RC

```
//---------------------------------------------------------------------------
// Circle.rc
//---------------------------------------------------------------------------

#define  NOCOMM
#include <windows.h>

#include "vsaver.h"


//---------------------------------------------------------------------------
// Control resources for CIRCLE control model
//---------------------------------------------------------------------------
IDBMP_CIRCLE        BITMAP  DISCARDABLE "circlecu.bmp"
IDBMP_CIRCLEDOWN    BITMAP  DISCARDABLE "circlecd.bmp"
IDBMP_CIRCLEMONO    BITMAP  DISCARDABLE "circlemu.bmp"
IDBMP_CIRCLEEGA     BITMAP  DISCARDABLE "circleeu.bmp"


//---------------------------------------------------------------------------
// Dialog to set flash color
//---------------------------------------------------------------------------
FlashDlg DIALOG LOADONCALL MOVEABLE DISCARDABLE 70, 18, 154, 94
    STYLE   DS_MODALFRAME | WS_CAPTION | WS_SYSMENU | WS_POPUP
    CAPTION "Flash Color"
    {
    CONTROL "OK",          DI_OK,      "button", BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD, 16, 73, 35, 14
    CONTROL "Cancel",      DI_CANCEL,  "button", BS_PUSHBUTTON    | WS_TABSTOP | WS_CHILD, 88, 72, 36, 14
    CONTROL "Red Color",   DI_REDOPT,  "button", BS_RADIOBUTTON   | WS_TABSTOP | WS_CHILD, 53, 10, 53, 12
    CONTROL "Green Color",DI_GREENOPT,"button", BS_RADIOBUTTON   | WS_TABSTOP | WS_CHILD, 53, 30, 58, 12
    CONTROL "Blue Color",  DI_BLUEOPT, "button", BS_RADIOBUTTON   | WS_TABSTOP | WS_CHILD, 53, 50, 47, 12
    }
```


Windows プログラミング用語集
コンストラクタ（Constractor）　オブジェクトのインスタンスを構築するための特殊な手続き。

特集
進化するWindowsの開発環境

2. ►►►►►►►►►►► C++化したWindowsの処理系

Windowsのような GUI に対応したプログラミングとオブジェクト指向の相性がよいことは，以前からよく知られていた．しかし，オブジェクト指向プログラミングを実践するためには，オブジェクト指向に対応した処理系と優れたクラスライブラリが必要となる．Micorsoft 社が発売を開始したMS-C/C++ 7.0は，オブジェクト指向言語であるC++に対応し，Windows 用クラスライブラリを標準装備した最新のWindows 開発環境をプログラマに提供する．また，Borland 社の Borland C++ & Application Frameworks 3.0も同様にC++に対応し，クラスライブラリを備えている．これら2つの処理系とクラスライブラリを用いたプログラミングの実際を紹介する．

WINDOWS Programming
特集 進化するWindowsの開発環境

Windowsの新処理系を概観する

Microsoft C/C++ Development System for Windows Version 7.0

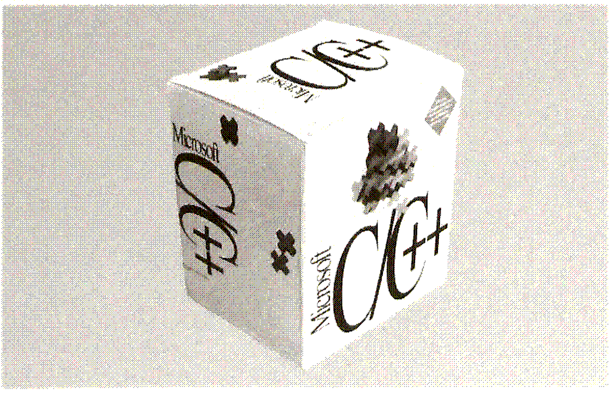
怡土昌彦

米国Microsoft社から，新しい処理系・MS-C/C++ 7.0が発売された．ほぼ同時期に，Windows Version 3.1も発売されている．当特集ではWindows 3.1との関連も含めて，この新しいMS-C/C++ 7.0について詳しく紹介する．

脅威の巨大パッケージ

MS-C 6.0の後継に位置づけられるこの新しい製品は，正式名称をMicrosoft C/C++ Development System for Windows Version 7.0（以後MS-C/C++ 7.0と略す）といい，名前が示す通りC++のサポートが目玉となっている．C++のサポートは，MS-C 6.0が発売される以前から何度となく噂に上ってはいたが，実際には読者も周知の通り実現されていなかった．その間に，Borland社からC++コンパイラが発売された．今回MS-CがC++に対応したことにより，より一層の競争が始まることであろう．

筆者の元に届いたMS-C/C++ 7.0のパッケージは，その巨大さで筆者を一気に圧倒した．もし店頭で購入するならば，車で行くか，店員に宅急便をお願いしたくなるであろう．まちがっても持ち帰るなどという無謀な真似は避けたくなるしろものである．なにしろ幅50cm，高さ21cm，奥行き28cmの大きさなのだ．ちょうど週刊少年漫画19冊分を重ねて横倒しにした感じである．そのうえ，重量は17kgもあるといわれている．読者も近くに比較できるものがあれば，その巨大さに目を見張ること請け合いである．



◀写真1　MS-C/C++ 7.0パッケージ

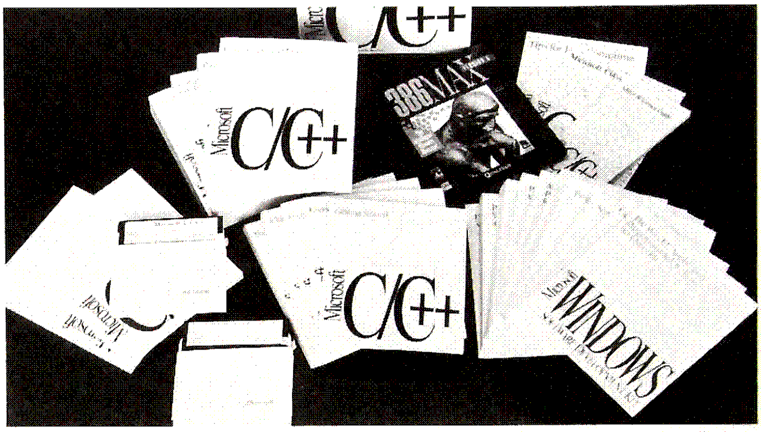
インストレーション

インストレーションには，少々覚悟が必要である．なにしろパッケージにはコンパイラとWindows 3.1のSDK，さらにプロファイラや386MAXまでが梱包されているからである．すべてをインストールするには，4回（場合によっては5回）もインストール作業を行わなければならない．

Windowsプログラミング用語集
コンテキストスイッチ（Context Switch）　Windowsにおいてタスク間のプロセッサ転換を果たすために必要な情報をセーブ・リストアすること．

これらが占有するディスク容量を考えると，気が遠くなる．

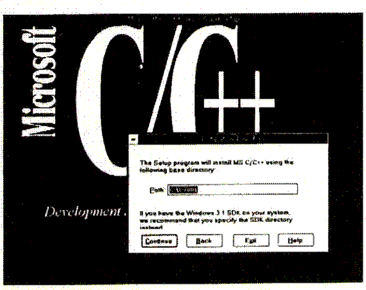
製品構成は，コンパイラが5インチ2HDのディスク11枚，関連マニュアルが9冊付属する．さらにWindows Software Development Kit（以後Windows SDK）は，ディスク12枚，マニュアル11冊．プロファイラがディスク1枚，マニュアル1冊．とどめに386MAXがディスク1枚，マニュアル2冊である．ほとんどお手上げの気分になってしまう．

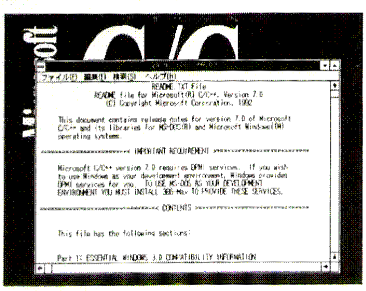

写真2　膨大なディスクとマニュアル

コンパイラ

コンパイラはインストールをWindowsから行うようになっている（画面1）．README.TXTがメモ帳で表示され，インストールしながら読めるようになっていて（画面2），たいへん便利なインストールプログラムに仕上がっている．

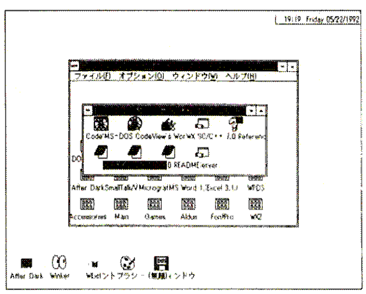
インストールが完了すると，プログラムマネージャにグループ登録が行われる．こんなところをみると，MS-C/C++ 7.0はWindowsのために開発されたんだな，という気になってくるから不思議である（画面3）．インストールがすべて終わると，図1のようにインストレーションされているはずである．

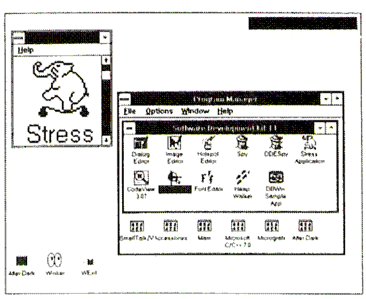

画面1


画面2

Windows SDK

コンパイラがWindowsからインストールするのに対して，SDKのインストールはなぜかDOSプロンプトから行わなければならない．DOS互換ボックスからのインストールでも，Windowsを終了するようにと叱られてしまう．SDKが対象としているWindowsは，バージョン3.1になっているが，オプションでリアルモード用プログラムを含め，バージョン3.0用のプログラムが開発できるようになっている．


画面3



画面4

SDKは3.0の時のそれとあまり大差はないようであるが，日本語版に3.1用SDKが付属するかどうか現時点では不明である（画面4）．なにしろ日本語Windows 3.1が存在しない内に3.1対応SDKが出荷されるとは思えないからである．この点に付いては，後日マイクロソフト社か各メーカーに問い合わせてみたいと思う．まちがってもSDKが付属しないという状況は考えられないので，最悪の場合はWindows 3.0のSDKと同等のものが付属し，後日差分なり何なりかが，何らかの方法で提供されてくるのではないだろうか？　どっちにしろ筆者のように，従来までのSDKを別途購入していた人間にとっては，なんだか損をしていたような気がするのは気のせいだろうか．

インストールがすべて終わると，図2のようにインストレーションされているはずである．

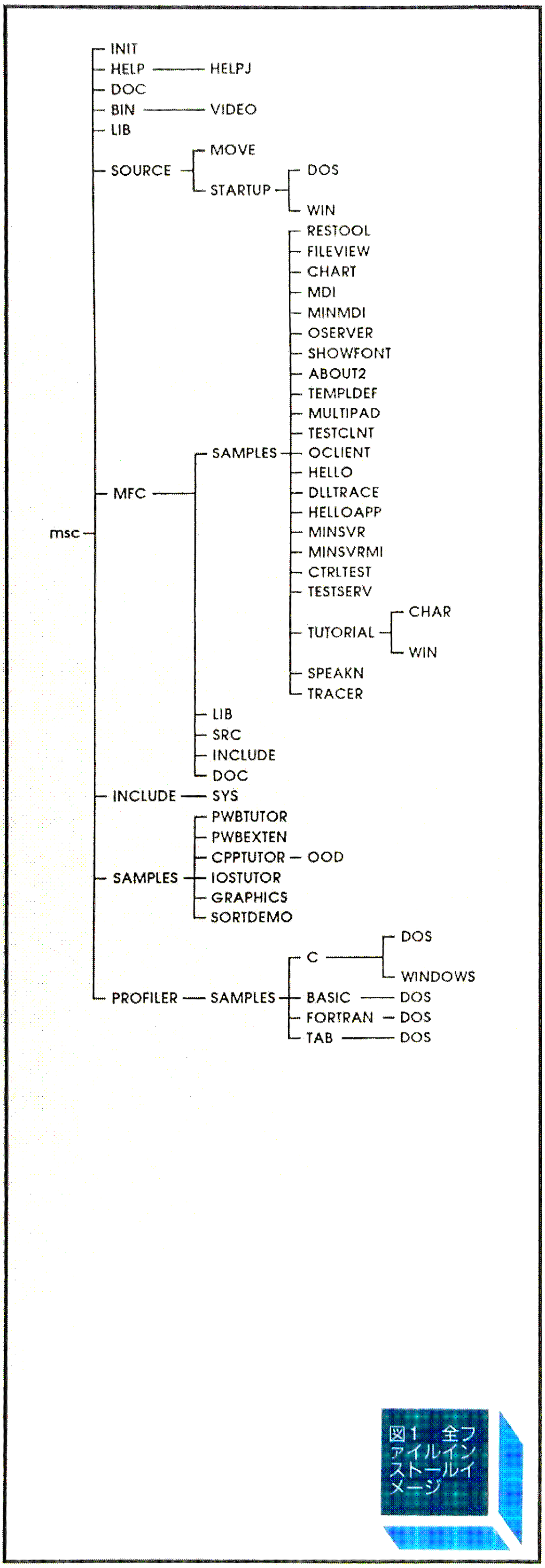


図1　全ファイルインストールイメージ

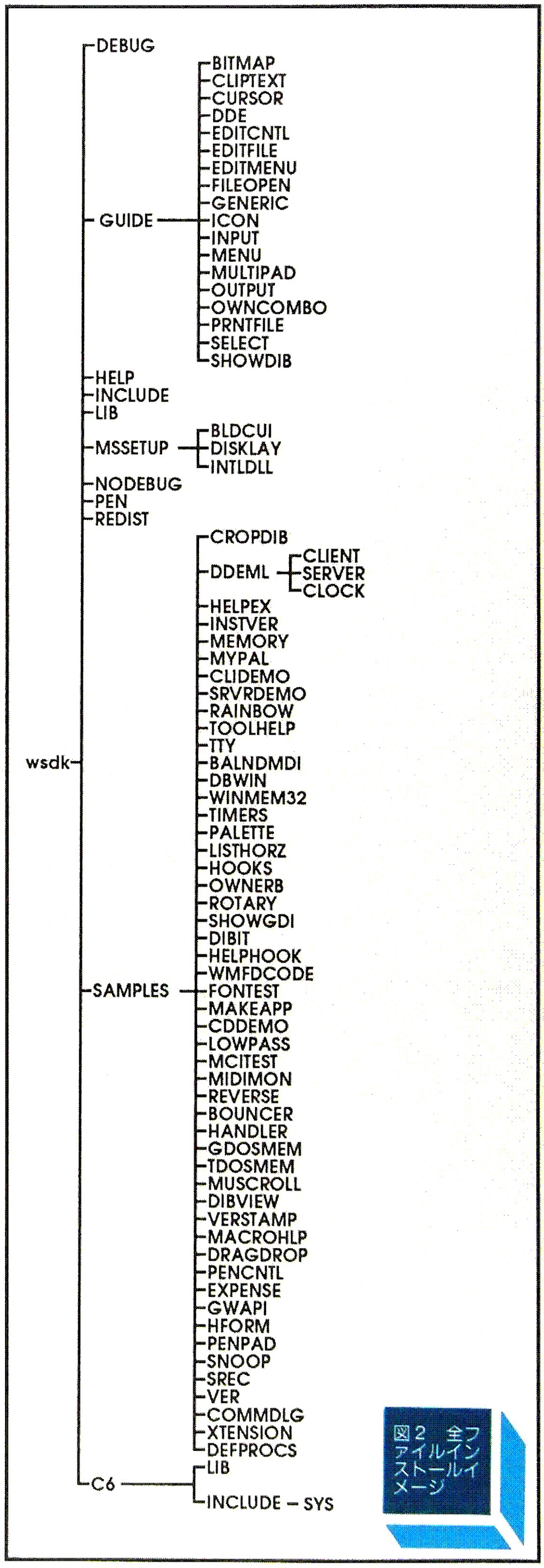


図2　全ファイルインストールイメージ

Windows プログラミング用語集
シェル（Shell）　別のアプリケーションを起動できるアプリケーションウィンドウのこと．プログラムマネージャやファイルマネージャがその例．

プロファイラ

プロファイラのインストレーションもDOSプロンプトから行うようになっている．インストールを行っていて気が付いたのだが，このツールだけが唯一OS/2に対応している．なお，このツールは単体でも発売されている．

386MAX

このプログラムのインストレーションもDOSプロンプトから行う．ただし，このプログラムだけはインストレーション直後にすぐ利用することはできず，しばらくはCONFIG.SYSと格闘することになる．どの程度格闘するかは，何をメモリに常駐させているかによって変わってくるが，かな漢などを利用している場合，その死闘は壮絶なものになるだろう．筆者はまる1日格闘して，DOS/Vで利用するのを断念したほどである．

しかし英語モードのみで利用するならば，630KB近くのメモリをDOSへ解放することができる．筆者のところでも629KB解放することができた．なお386MAXを組み込んで，Windowsを利用する場合はSYSTEM.INIに変更を加える必要がある（Windows 3.1の場合は，386MAX自体の設定にも変更が必要）．それらの詳細については，386MAXのマニュアルやWindowsのドキュメントに記述されている．

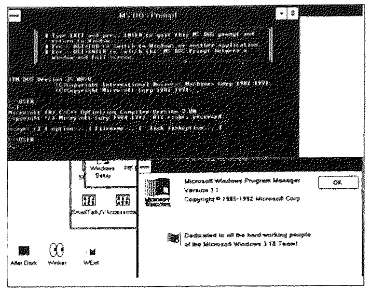
MS-C 6.0 ライブラリ

今までMS-C 6.0+3.0用SDKでWindowsプログラムを作成していたユーザーは，MS-C/C++ 7.0に移行しない限りWindows 3.1用プログラムを作成することはできない．なぜなら，3.1で新しく追加されたWindows APIが，3.0用SDKには用意されていないからである．

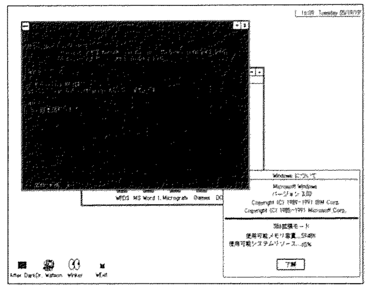
しかしSDKのほうには，MS-C 6.0互換のライブラリとインクルードファイルが用意されており，それらを利用するとMS-C 6.0でもWindows 3.1用プログラムを作成できるようになっている．

32ビットコンパイラ

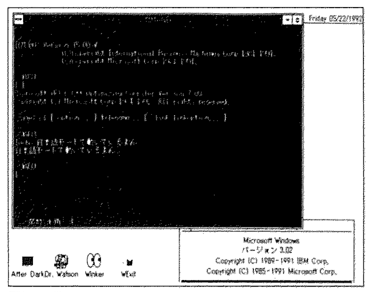
MS-C/C++ 7.0もついにIBM PC以外では動作しなくなった．従来のバージョンでも，CodeViewデバッガなどは機種依存していたのだが，コンパイラ自身の動作は可能であった．しかし，今度のコンパイラは動作しない．もっと正確に伝えると，たとえIBM PCやその互換機でもiAPX 286（80286）以下のCPUを搭載しているマシンでは動作しないのである．なにしろ，コンパイラ自身が32ビットの実行コードで記述されているネイティブプロテクトモードコンパイラなのである．このため，コンパイラを動作させるには，DOSエクステンダ(DOS EXTENDER)と呼ばれるメモリマネージメントを行うプログラムが必須となり，32ビットDOSエクステンダが動作しない286以下の機種では動作できないのである．



画面5



画面6



画面7

もっとも，コンパイラ自身は32ビットに依存はしているが，機種には依存していないので，コンパイラが対応したDOSエクステンダさえ動作する環境があればIBM系以外のマシンでも動作させることが可能である．コンパイラが対応しているDOSエクステンダは，DPMI規格に準拠しているものに限られている．もちろんMS-C/C++ 7.0には，386MAXと呼ばれるメモリマネージメントプログラムがバンドリングされているので，ユーザーはDOSエクステンダを別途購入することなく，コンパイラを動作させることが可能になっている．

DPMIといえば勘のいい読者なら

Windowsプログラミング用語集
スコープ（Scope）　適用範囲のこと，識別子のスコープは，その識別子をオブジェクトにアクセスするために使用できるプログラムの一部分．

気が付くと思うが，WindowsのDOS互換ボックスがサポートしている規格である．事実MS-C/C++ 7.0は，386MAXをインストールしないならばDOS互換ボックスを利用するようにいってくる．これを試さずしてレポートは書けないと思い，日本語Windows 3.0（IBM版／NEC版），英語版Windows 3.0および英語版Windows 3.1において動作確認テストを行ってみた．結果は日本語Windows以外は正常に動作した（画面5，6）．しかし，日本語WindowsでもIBM版に限りSYSTEM.INIを書き換えることにより動作させることができるようになった（図3，画面7）．ただし，日本語Windows 3.0下での動作をマイクロソフト社は保証していない．ときどきではあるが，筆者のマシンでもUAEが発生している．

```
system.ini
[386Enh]
device=C:¥MSC¥BIN¥vmcpd.386      <----- 絶対パスで追加
device=C:¥MSC¥BIN¥vpfd.386       <----- 絶対パスで追加
device=C:¥MSC¥BIN¥cvw1.386       <----- 絶対パスで追加
device=C:¥MSC¥BIN¥vmb.386        <----- 絶対パスで追加

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

device=*parity
device=*biosxlat
device=*vcd
;device=*vmcpd                <----- コメントにする。
device=*combuff
device=*cdpscsi
```

図3　IBM版ウィンドウズのパッチ方法



MS-C/C++ 7.0には，統合環境としてProgrammer's WorkBench（PWB）の最新バージョン2.0が付属している（画面8）．旧バージョンのPWBはディスクスワップが頻繁に発生し，お世辞にも速いとはいえなかった．しかし今度のPWBは，かなりの機能アップが行われている．

(1)高速な動作

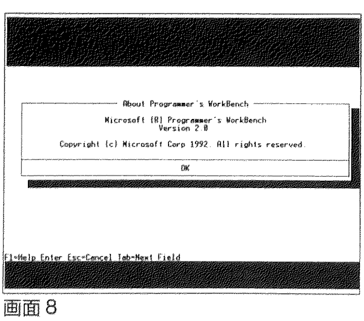
まず，何といっても立ち上がりやコンパイルへの移行が高速である．Windowsから利用しても，ストレスを感じない程である．この理由にはXMSのサポートもあるが，かなりのブラッシュアップも行われたと思う．これなら簡単なプログラムや，makefileのスケルトンを作成するのであれば，PWBで行ってしまったほうが簡単に，高速に作業ができると思えたほどだ．なぜならPWBのプロジェクトは，ターゲットプログラム環境を指定すれば，最良のコンパイラオプションを選択してくれるからである．なによりも，大量にあるコンパイラやリンカのオプションをいちいち調べて選択するよりは，PWBが作ったmakefileに手を加えたほうが楽である．趣味でプログラミングを行っている人ならば，エディタ要らずになるであろう．

(2)オーバーラップウィンドウ

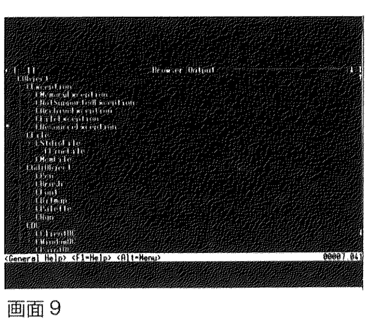
2番目に気がつく大きな変更は，ウィンドウのスタイルが変更されている点である．従来まで，PWBのウィンドウはタイリング表示されていたが，今回のバージョンからオーバーラップ表示に変更されている．タイル表示とオーバーラップ表示のどちらがよいかは別にして，Windowsと感覚的に操作性が同じになった点は嬉しい限りである．

(3)クラスをツリー表示

当然のことながら，ブラウザ類がC++に対応してクラスの階層をツリー表示（Class Hierarchy）できるようになっている（画面9）．従来のバージョンより，検索，表示の切り換えなどが高速に行われるようになっていることはいうまでもない．1つ残念な点は，しかたがないことなのだが，コンパイルを正常に完了させるまでブラウザを利用できないことである．いずれ，



画面8



画面9

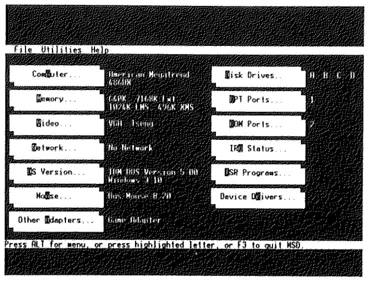
Windowsプログラミング用語集
スタック（Stack）　プログラムの関数へ渡された引数を保持する領域　サイズは変化しないがデータの量はプログラム実行中に変化する．

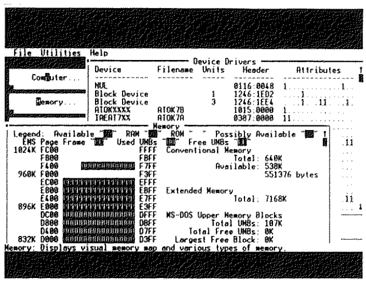
この点に対処したバージョンを出荷してもらえることを希望したい.

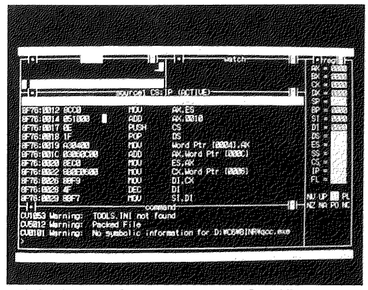
(4)ウィンドウのアイコン化

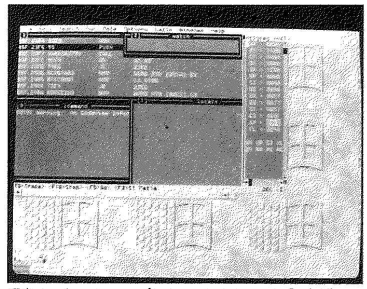
おもしろい変更は，それぞれのウィンドウ（ダイアログは除く）をアイコンに変化させることができる点である（画面10）．むろんWindowsのようにイメージ的な絵がでるわけではないが，しばらく使わないウィンドウをアイコン化しておくことができて便利である．

PWBには，.PIFファイルが付属しているので，Windowsから直接呼び出してプログラム開発を行えるようになっている．


画面11


画面12


写真3　タイリングウィンドウを用いたCodeView

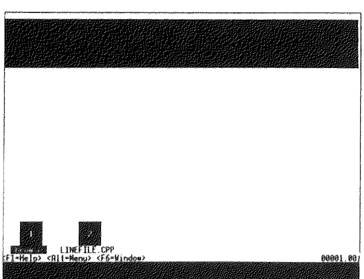

写真4　オーバーラップドウィンドウを用いたCodeView

NMAKE

メイクプログラムもかなり機能アップされている．従来までのNMK.COMがなくなり，NMAKEで大規模なプログラムがコンパイルできるようになったからである．NMAKEもXMSをサポートしており，コンパイラに十分なメモリを提供するようになった点もユーザーにとっては嬉しい限りである．

DIAGNOSTICS

DIAGNOSTICSは，ユーザーのコンピュータ環境をチューンナップするためのツールである．このプログラムを利用すると，現在のメモリ利用状況，プログラムの常駐状況などをつぶさに見ることができる(画面11，12)．ユーザーは，このプログラムの出力を見てCONFIG.SYSやAUTOEXEC.BATをカスタマイズしていくことになる．Quarterdeck社のQEMMや，Qualitas社の386MAXを利用したことのあるユーザーなら，それらに付属しているMFT.EXEやASQ.EXEみたいなものだと思えばよい．


画面10

MS-DOS 5.0になって，ハイメモリ領域を駆使するようになっている現在では，メモリマネージメントソフトを購入しなくても利用できるプログラムをユーザーは希望していたのではないだろうか．筆者もこのプログラムの存在は前々から知ってはいたのだが，実物を見たのは初めてであった．

CodeView

CodeView（CV）は，MS-Cユーザーお馴染みのソースコードデバッガである．バージョンも4.0に上がり，ウィンドウがPWBと同じようにオーバーラップウィンドウに変更されている．PWBのオーバーラップウィンドウ化には，あまり便利さを感じなかったのだが（筆者が今までPWBを使いこなしていないこともあるが……），CVでのオーバーラップウィンドウ化はたいへん便利であった．なにしろウィンドウが重なるので，必要なウィンドウが広いのである．たとえば，Local Watch，Global Watch，Source，Registerの各ウィンドウを開いていたとしよう．従来のCVでは写真3のようになってしまい，よく利用するWatchウィンドウとSourceウィンドウの表示領域が狭くなってしまっていた．これが，オーバーラップウィンドウになると，写真4のようになり広々と利用できるのである．

画面3を注意して見ると，CodeViewのアイコンが2つ存在していることに気がつく．1つは，CodeView for Windowsではないか？　と思われた

方は，画面4も一緒に見てもらいたい．CodeView for Windowsは，SDKのほうに別途用意されているのである．なぜ3つもCVが提供されているのか，筆者なりにいろいろと調べた結果，次のようなことではないかと結論づけた．まず，

(1) SDKに付いているCVだが，このCVは古いバージョン(3.07)が付属している．そして，MS-Cのほうには，4.0のDOS版，Windows版が付属している．

(2)今までSDKは単体で売られていて，それらはMS-C 6.0に対応しているものであった．そして今回のMS-C/C++ 7.0には，そのSDKがそのまま付属した形で販売されたので，古いCVが付属している．

(3) MS-C/C++ 7.0は前にも述べたとおり，iAPX80386以降のCPUでないと動作しない．しかしWindowsプログラム開発者が全員386機種を持っているとは限らず，それらの人々は従来通りMS-C 6.0を使い続けることになるだろう．しかしMS-C 6.0で使える3.0用SDKには追加されたAPIのライブラリは付属していないので，MS-C 6.0を使い続ける開発者は，3.1用のプログラムが開発できなくなってしまう．それらを避けるために，追加APIを含んだMS-C 6.0互換ライブラリと，古いCVが付属している．

もちろん，MS-C/C++ 7.0のほうに付属しているCVであればC++に対応している点はいうまでもない．しかしSDKに付属しているCVは，MS-C/C++ 7.0で開発したWindowsアプリケーションをデバッグすることはできないようである．

新しいCVでは，今まで望まれていたリモートデバッグもサポートされている．リモートデバッグが可能になれば，今までデバッガを使うと再現されにくいバグも発見しやすくなる．

WX-Server

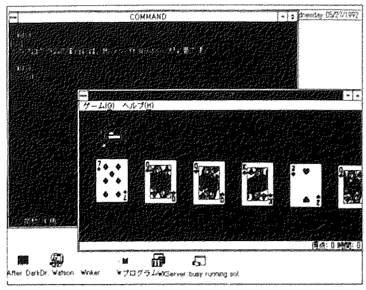
DOS互換ボックスからWindowsプログラムを起動すると，本来なら「このプログラムを実行するにはWindowsが必要です．」と警告されるのだが，WX-Serverを利用すると起動することができるようになる（画面13）．これにより，WindowsからPWBを呼び出して開発，そのまま実行／デバッグが可能になった．今までプログラムができたらWindowsに戻り，ファイルマネージャなどで目的プログラムを探し出して実行していたことを考えるとたいへん便利なツールである．

マウス

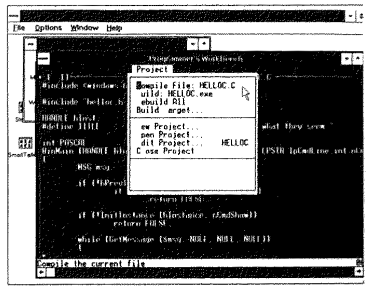
MS-C/C++ 7.0には，新しいマウスドライバが付属している．このマウスドライバは，なんとウィンドウに閉じこめたDOSアプリケーションのマウス操作も可能なのである（画面14）．ただしそれができるのは，今のところWindows 3.1だけである．

コンパイラ

コンパイラは，先に述べた32ビット化以外にも各種の拡張が行われている．たとえば，



画面13



画面14

(1)ヘッダーコンパイル機能

インクルードファイルだけをコンパイルして中間コード化（プリコンパイルヘッダ）しておき，メインプログラムのコンパイル時間を短縮する．Windows用ヘッダファイルのように巨大なヘッダファイルをコンパイルする場合や，複数のヘッダを大量にインクルードしなければならない場合に効果を発揮する．

注目する点は，任意にプリコンパイルヘッダファイル名を付けることができるために，ヘッダファイルの組み合わせに応じたプリコンパイルヘッダを自由に作成できる．したがって，ヘッダの順番が変わっても影響がないように考慮されている．

(2)最適化機能の強化

最適化の強化は，特に力が入れられているようである．まずインライン展開．関数のインライン展開といえばC++で提供された機能で，Cで実現するには#defineを使ったマクロの利用しか方法がなかった．しかし，この機能がCでサポートされたことによ

Windows プログラミング用語集
スプーラ（Spooler）　マルチタスクシステムにおいて複数のタスクからのプリント要求をスタックする機能．同時にプリントデータのバッファリングも行う．

り，より柔軟にプログラムを書けるようになった．

次にP-Codeのサポート．これは本来マイクロソフト社内でのみ利用していた機能で，従来のマシンコード形式とは違う仮想マシン用のマシンコードを生成する．この仮想マシンコードを使うと，今まで数ステップ必要だったCPU命令を1ステップで完了するようになる．このためコードサイズが格段に圧縮され，プログラムを小さくすることができるようになる訳だ（ただし使い方を誤れば返って大きくなることもありえる）．しかしこのP-Code，仮想マシンが相手だけに従来の8086系マシンで直接実行することができない．したがって，実行時にはインタプリタが必要になり，実行速度は低下する．ただし，従来のマシンコードとの併用が可能になっているため，うまく組み合わせれば，それほど実行速度を低下させずにコード，サイズ共に最適化ができるようになる．

P-Codeは，Windowsプログラムでも利用可能になっている．実際にP-Codeを使った製品として，Excel 3.1やMS-Worksなどがある．

参考までに，実験に利用したソースコードと-Faオプションによって吐き出された.ASMファイルをリスト1，2，3に掲載しておく（なお，すべての最適化は-Odオプションによって禁止してある）．これをみても，P-Codeは利用の仕方によって薬にも毒にもなる機能だということがわかる．

(3) C++ Version 2.1規格

今までに何度となくC++をサポートしていると紹介しているが，MS-C/C++ 7.0ではAT&T C++ Ver.2.1規格をサポートしている．しかし，Ver2.1規格を完全にサポートしているわけではなく，型を与えてクラスを生成するテンプレート機能などはサポートされていない．しかし，疑似的にテンプレートを利用できるようにするプログラムが，サンプルとして付属している．これらの問題は，C++のANSI規格が発表される頃には解決するのではないだろうか．簡単なC++のサンプルプログラムを作成したので掲載しておく（リスト4）．

(4)ネイティブコードのサポート（予約のみ）

ネイティブプロテクトモード用のコードを生成するオプションが，予約だけではあるがサポートされた．これは，近い将来リリースされるWindows NT（WIN32）のための予約オプションである．このオプションが使える日が来るのも遠くないであろう．

リスト 1 サンプルに利用したプログラム

```
/*
 $Id$
 $Copyright (c) 1992. Sazaby's Inc.  All rights reserved.$
 $Release (R) 1992. Sazaby's Inc.  (w)Masahiko Ito.$
 *
 $Log$
 *
 */
#include <ctype.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void ShowFreeBytes (void);
char szUnit[10] = "¥0¥0";
unsigned long freeBytes;
unsigned DriveNo = 0;

void
ShowHelp (void)
{
    printf ("Free.exe  Version 1.3 for MS-DOS¥n");
    printf ("Copyright (c) 1990-91  Sazaby's.  All rights reserved.¥n¥n");
    printf ("free はディスクの空き容量を B, KB, MB のいずれかで表¥示します.¥n");
    printf ("Usage: free [-b|-k|-m] [d:]¥n");
    exit (1);
}


void
GetOptions (char* argvx[])
{
    int k, ret;
    char szDrive[5];

    k = 1;
    if (argvx[k][0] == '-' || argvx[k][0] == '/')
    {
        if (argvx[k][2] != '¥0')
        {
            printf ("Unknown option.¥n");
            exit (1);
        }
        switch (argvx[k][1])
        {

          case 'k':
          case 'K':
            strcpy (szUnit, "KBytes");
            break;

          case 'm':
          case 'M':
            strcpy (szUnit, "MBytes");
            break;

          case '?':
            ShowHelp ();;
            break;

          case 'b':
          case 'B':
            strcpy (szUnit, "Bytes");
            break;

          default :
            printf ("Unknown option.¥n");
            exit (1);
        }

        k++;
    }

    if (szUnit[0] == '¥0')
        strcpy (szUnit, "Bytes");
    strncpy (szDrive, &argvx[k][0], 3);
    szDrive[2] = '¥0';
    ret = SetDrive (&argvx[k][0]);
    if (ret < 0)
    {
        printf ("Unknown drive.¥n");
        exit (1);
    }
}


int
SetDrive (char chDrive[])
{

    if (chDrive[0] != '¥0')
    {
        if (!((chDrive[0] >= 'A' && chDrive[0] <= 'Z') ||
              (chDrive[0] >= 'a' && chDrive[0] <= 'z')))
            return -1;

        if (chDrive[1] != ':')
            return -1;

        if (chDrive[2] != '¥0')
```

Windowsプログラミング用語集
スペーシング(Spacing)　テキストをディスプレイに表示する行送り．Windowsはテキストもビットマップで表示するのでスペーシングサイズは固定されていない．

```
            return  -1;
        DriveNo = toupper (chDrive[0]) - '@';
    }
    else
        _dos_getdrive (&DriveNo);
    return 0;
}


main (int argc, char* argv[])
{
    struct _diskfree_t dtable;
    int     ret;

    if (argc > 3)
        ShowHelp ();
    if (argc > 1)
        GetOptions (argv);
    else
    {
        strcpy (szUnit, "Bytes");
        ret = SetDrive (NULL);
        if (ret < 0)
        {
            printf ("Unknown drive.¥n");
            exit (1);
        }
    }

    _dos_getdiskfree (DriveNo, &dtable);
    freeBytes =
        (unsigned long) (dtable.bytes_per_sector *
                         dtable.sectors_per_cluster) *
        (unsigned long) dtable.avail_clusters;

    ShowFreeBytes ();
    return 0;
}


void
ShowFreeBytes (void)
{
    if (!strcmp (szUnit, "KBytes"))
        printf ("%ld", (freeBytes / 1024UL));
    else if (!strcmp (szUnit, "MBytes"))
        printf ("%ld", (freeBytes / (1024UL * 1024UL)));
    else
        printf ("%ld", freeBytes);
    printf ("%s free.¥n", szUnit);
}

```

リスト 2　-Od -Fa オプションによってコンパイラが生成したコード

```
; File free.c
; Line 14
_ShowHelp:
        push    bp
        mov     bp, sp
        mov     ax, OFFSET L00494
        call    __aNchkstk
        push    si
        push    di
; Line 15
        mov     ax, OFFSET L00426
        push    ax
        call    _printf
        add     sp, OFFSET 2
; Line 16
        mov     ax, OFFSET L00427
        push    ax
        call    _printf
        add     sp, OFFSET 2
; Line 17
        mov     ax, OFFSET L00428
        push    ax
        call    _printf
        add     sp, OFFSET 2
; Line 18
        mov     ax, OFFSET L00429
        push    ax
        call    _printf
        add     sp, OFFSET 2
; Line 19
        mov     ax, OFFSET 1
        push    ax
        call    _exit
        add     sp, OFFSET 2
; Line 20
; Line 20
L00425:
        pop     di
        pop     si
        mov     sp, bp
        pop     bp
        ret     OFFSET 0
; Line 0
; Line 25
; argvx = 0004
_GetOptions:
        push    bp
        mov     bp, sp
        mov     ax, OFFSET L00496
        call    __aNchkstk
        push    si
        push    di
; k = fffc
; ret = fffa
; szDrive = fff4
; Line 26
; Line 27
; Line 29
        mov     WORD PTR -4[bp], OFFSET 1
; Line 30
        mov     ax, WORD PTR -4[bp]
        shl     ax, OFFSET 1
        mov     bx, WORD PTR 4[bp]
        add     bx, ax
        mov     bx, WORD PTR [bx]
        cmp     BYTE PTR [bx], OFFSET 45
        je  L00438
        mov     ax, WORD PTR -4[bp]
        shl     ax, OFFSET 1
        mov     bx, WORD PTR 4[bp]
        add     bx, ax
        mov     bx, WORD PTR [bx]
        cmp     BYTE PTR [bx], OFFSET 47
        jne     L00437
L00438:
; Line 31
; Line 32
        mov     ax, WORD PTR -4[bp]
        shl     ax, OFFSET 1
        mov     bx, WORD PTR 4[bp]
        add     bx, ax
        mov     bx, WORD PTR [bx]
        cmp     BYTE PTR 2[bx], OFFSET 0
        je  L00439
; Line 33
; Line 34
        mov     ax, OFFSET L00440
        push    ax
        call    _printf
        add     sp, OFFSET 2
; Line 35
        mov     ax, OFFSET 1
        push    ax
        call    _exit
        add     sp, OFFSET 2
; Line 36
; Line 37
L00439:
        mov     ax, WORD PTR -4[bp]
        shl     ax, OFFSET 1
        mov     bx, WORD PTR 4[bp]
        add     bx, ax
        mov     bx, WORD PTR [bx]
        mov     al, BYTE PTR 1[bx]
        cbw
        jmp     L00441
; Line 38
; Line 40
L00445:
; Line 41
; Line 42
        mov     ax, OFFSET L00446
        push    ax
        mov     ax, OFFSET _szUnit
        push    ax
        call    _strcpy
        add     sp, OFFSET 4
; Line 43
        jmp     L00442
; Line 45
L00447:
; Line 46
; Line 47
        mov     ax, OFFSET L00448
        push    ax
        mov     ax, OFFSET _szUnit
        push    ax
        call    _strcpy
        add     sp, OFFSET 4
; Line 48
        jmp     L00442
; Line 50
L00449:
; Line 51
        call    _ShowHelp
; Line 52
        jmp     L00442
; Line 54
L00450:
; Line 55
; Line 56
        mov     ax, OFFSET L00451
        push    ax
        mov     ax, OFFSET _szUnit
        push    ax
        call    _strcpy
        add     sp, OFFSET 4
; Line 57
        jmp     L00442
; Line 59
L00452:
; Line 60
        mov     ax, OFFSET L00453
```


Windows プログラミング用語集
スラッシング（Thrashing）　マルチタスクでほかのタスクのコードをディスクからロードする時間を指す.

```
            push    ax
            call     _printf
            add     sp,OFFSET 2
; Line 61
            mov     ax,OFFSET 1
            push    ax
            call     _exit
            add     sp,OFFSET 2
; Line 62
; Line 62
            jmp     L00442
L00441:
            cmp     ax,OFFSET 63
            je  L00449
            cmp     ax,OFFSET 66
            je  L00450
            cmp     ax,OFFSET 75
            je  L00445
            cmp     ax,OFFSET 77
            je  L00447
            cmp     ax,OFFSET 98
            je  L00450
            cmp     ax,OFFSET 107
            je  L00445
            cmp     ax,OFFSET 109
            je  L00447
            jmp     L00452
L00442:
; Line 64
            add     WORD PTR -4[bp],OFFSET 1
; Line 65
; Line 67
L00437:
            cmp     BYTE PTR _szUnit,OFFSET 0
            jne     L00454
; Line 68
            mov     ax,OFFSET L00455
            push    ax
            mov     ax,OFFSET _szUnit
            push    ax
            call     _strcpy
            add     sp,OFFSET 4
; Line 69
L00454:
            mov     ax,OFFSET 3
            push    ax
            mov     ax,WORD PTR -4[bp]
            shl     ax,OFFSET 1
            mov     bx,WORD PTR 4[bp]
            add     bx,ax
            push     WORD PTR [bx]
            lea     ax,WORD PTR -12[bp]
            push    ax
            call     _strncpy
            add     sp,OFFSET 6
; Line 70
            mov     BYTE PTR -10[bp],OFFSET 0
; Line 71
            mov     ax,WORD PTR -4[bp]
            shl     ax,OFFSET 1
            mov     bx,WORD PTR 4[bp]
            add     bx,ax
            push     WORD PTR [bx]
            call     _SetDrive
            add     sp,OFFSET 2
            mov     WORD PTR -6[bp],ax
; Line 72
            cmp     WORD PTR -6[bp],OFFSET 0
            jge     L00457
; Line 73
; Line 74
            mov     ax,OFFSET L00458
            push    ax
            call     _printf
            add     sp,OFFSET 2
; Line 75
            mov     ax,OFFSET 1
            push    ax
            call     _exit
            add     sp,OFFSET 2
; Line 76
; Line 77
L00457:
; Line 77
L00433:
            pop     di
            pop     si
            mov     sp,bp
            pop     bp
            ret     OFFSET 0
; Line 0
; Line 82
; chDrive = 0004
_SetDrive:
            push    bp
            mov     bp,sp
            mov     ax,OFFSET L00497
            call     __aNchkstk
            push    si
            push    di
; Line 84
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR [bx],OFFSET 0
            je  L00462
; Line 85
; Line 86
; Line 87
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR [bx],OFFSET 65
            jl  L00464
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR [bx],OFFSET 90
            jle     L00463
L00464:
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR [bx],OFFSET 97
            jl  L00465
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR [bx],OFFSET 122
            jle     L00463
L00465:
; Line 88
            mov     ax,OFFSET -1
            jmp     L00461
; Line 90
L00463:
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR 1[bx],OFFSET 58
            je  L00466
; Line 91
            mov     ax,OFFSET -1
            jmp     L00461
; Line 93
L00466:
            mov     bx,WORD PTR 4[bp]
            cmp     BYTE PTR 2[bx],OFFSET 0
            je  L00467
; Line 94
            mov     ax,OFFSET -1
            jmp     L00461
; Line 95
L00467:
            mov     bx,WORD PTR 4[bp]
            mov     al,BYTE PTR [bx]
            cbw
            mov     bx,OFFSET __ctype+1
            add     bx,ax
            mov     al,BYTE PTR [bx]
            and     ax,OFFSET 255
            and     ax,OFFSET 2
            cmp     ax,OFFSET 0
            je  L00499
            mov     bx,WORD PTR 4[bp]
            mov     al,BYTE PTR [bx]
            cbw
            add     ax,OFFSET -32
            jmp     L00498
L00499:
            mov     bx,WORD PTR 4[bp]
            mov     al,BYTE PTR [bx]
            cbw
L00498:
            add     ax,OFFSET -64
            mov     WORD PTR _DriveNo,ax
; Line 96
; Line 97
            jmp     L00468
L00462:
; Line 97
; Line 98
            mov     ax,OFFSET _DriveNo
            push    ax
            call     __dos_getdrive
            add     sp,OFFSET 2
L00468:
; Line 99
            mov     ax,OFFSET 0
            jmp     L00461
; Line 100
; Line 100
L00461:
            pop     di
            pop     si
            mov     sp,bp
            pop     bp
            ret     OFFSET 0
; Line 0
; Line 104
; argc = 0004
; argv = 0006
_main:
            push    bp
            mov     bp,sp
            mov     ax,OFFSET L00500
            call     __aNchkstk
            push    si
            push    di
; dtable = fff6
; ret = fff4
; Line 105
; Line 106
; Line 108
            cmp     WORD PTR 4[bp],OFFSET 3
            jle     L00477
; Line 109
            call     _ShowHelp
; Line 110
L00477:
            cmp     WORD PTR 4[bp],OFFSET 1
            jle     L00478
; Line 111
            push     WORD PTR 6[bp]
            call     _GetOptions
            add     sp,OFFSET 2
; Line 112
            jmp     L00479
L00478:
; Line 112
```

```
; Line 113
; Line 114
        mov     ax,OFFSET L00480
        push    ax
        mov     ax,OFFSET _szUnit
        push    ax
        call    _strcpy
        add     sp,OFFSET 4
; Line 115
        mov     ax,OFFSET 0
        push    ax
        call    _SetDrive
        add     sp,OFFSET 2
        mov     WORD PTR -12[bp],ax
; Line 116
        cmp     WORD PTR -12[bp],OFFSET 0
        jge     L00481
; Line 117
; Line 118
        mov     ax,OFFSET L00482
        push    ax
        call    _printf
        add     sp,OFFSET 2
; Line 119
        mov     ax,OFFSET 1
        push    ax
        call    _exit
        add     sp,OFFSET 2
; Line 120
; Line 121
L00481:
L00479:
; Line 123
        lea     ax,WORD PTR -10[bp]
        push    ax
        push    WORD PTR _DriveNo
        call    __dos_getdiskfree
        add     sp,OFFSET 4
; Line 124
; Line 125
; Line 126
; Line 127
        mov     ax,WORD PTR -8[bp]
        mov     dx,OFFSET 0
        push    dx
        push    ax
        mov     ax,WORD PTR -4[bp]
        imul    WORD PTR -6[bp]
        mov     dx,OFFSET 0
        push    dx
        push    ax
        call    __aNulmul
        mov     WORD PTR _freeBytes,ax
        mov     WORD PTR _freeBytes+2,dx
; Line 129
        call    _ShowFreeBytes
; Line 130
        mov     ax,OFFSET 0
        jmp     L00474
; Line 131
; Line 131
L00474:
        pop     di
        pop     si
        mov     sp,bp
        pop     bp
        ret     OFFSET 0
; Line 0
; Line 136
_ShowFreeBytes:
        push    bp
        mov     bp,sp
        mov     ax,OFFSET L00502
        call    __aNchkstk
        push    si
        push    di
; Line 137
        mov     ax,OFFSET L00485
        push    ax
        mov     ax,OFFSET _szUnit
        push    ax
        call    _strcmp
        add     sp,OFFSET 4
        cmp     ax,OFFSET 0
        jne     L00484
; Line 138
        mov     ax,OFFSET 1024
        mov     dx,OFFSET 0
        push    dx
        push    ax
        push    WORD PTR _freeBytes+2
        push    WORD PTR _freeBytes
        call    __aNuldiv
        push    dx
        push    ax
        mov     ax,OFFSET L00486
        push    ax
        call    _printf
        add     sp,OFFSET 6
; Line 139
        jmp     L00487
L00484:
; Line 139
        mov     ax,OFFSET L00489
        push    ax
        mov     ax,OFFSET _szUnit
        push    ax
        call    _strcmp
        add     sp,OFFSET 4
        cmp     ax,OFFSET 0
        jne     L00488
; Line 140
        mov     ax,OFFSET 0
        mov     dx,OFFSET 16
        push    dx
        push    ax
        push    WORD PTR _freeBytes+2
        push    WORD PTR _freeBytes
        call    __aNuldiv
        push    dx
        push    ax
        mov     ax,OFFSET L00490
        push    ax
        call    _printf
        add     sp,OFFSET 6
; Line 141
        jmp     L00491
L00488:
; Line 141
; Line 142
        push    WORD PTR _freeBytes+2
        push    WORD PTR _freeBytes
        mov     ax,OFFSET L00492
        push    ax
        call    _printf
        add     sp,OFFSET 6
L00491:
L00487:
; Line 143
        mov     ax,OFFSET _szUnit
        push    ax
        mov     ax,OFFSET L00493
        push    ax
        call    _printf
        add     sp,OFFSET 4
; Line 144
; Line 144
L00483:
        pop     di
        pop     si
        mov     sp,bp
        pop     bp
        ret     OFFSET 0
; Line 0
```

リスト 3 -Oq -Od -Faオプションによってコンパイラが生成したコード

```
; Line 14
        BeginProc       native near cdecl ShowHelp
        Chkstk          0000
; Line 15
        LdngA3          $s(0000)
        CallS0          printf
; Line 16
        LdngA2          $s(0022)
        CallS0          printf
; Line 17
        LdngA1          $s(005b)
        CallS0          printf
; Line 18
        LdngA0          $s(009a)
        CallS0          printf
; Line 19
        LdcW1
        CallS1          exit
; Line 20
L00425:
        Ret
        EndProc         ShowHelp
; argvx = 0004
; k = fffc
; ret = fffa
; szDrive = fff4
; Line 25
        BeginProc       native near cdecl GetOptions
        Chkstk          0000
; Line 29
        LdcW1
        StfW0           k
; Line 30
        LdpW0           argvx
        LdfW0           k
        LdinoW
        LdinB0
        CvtBW
        LdcWub          2d
        JeqWb           L00438
        LdpW0           argvx
        LdfW0           k
        LdinoW
        LdinB0
        CvtBW
        LdcWub          2f
        JneWb           L00437
L00438:
; Line 31
```


Windows プログラミング用語集
ダイアログボックス（Dialog boxes）　チャイルドウィンドウの一種で，おもにユーザーからの入力を受け付けるのに使われる。

```
; Line 32
        LdpW0           argvx
        LdfW0           k
        LdinoW
        LdinB2
        JzrW5           L00439
; Line 33
; Line 34
        LdngAb          $s(00c4)
        CallS2          printf
; Line 35
        LdcW1
        CallS1          exit
; Line 36
; Line 37
L00439:
        LdpW0           argvx
        LdfW0           k
        LdinoW
        LdinB1
        CvtBW
        Gotob           L00441
; Line 38
; Line 40
L00445:
; Line 42
        LdngAb          $s(00d5)
        LdngA0          szUnit
        CallS0          strcpy
; Line 43
        PopIP           L00442
; Line 45
L00447:
; Line 47
        LdngAb          $s(00dc)
        LdngA0          szUnit
        CallS0          strcpy
; Line 48
        PopIP           L00442
; Line 50
L00449:
; Line 51
        CallS4          ShowHelp
; Line 52
        PopIP           L00442
; Line 54
L00450:
; Line 56
        LdngA4          $s(00e3)
        LdngA0          szUnit
        CallS0          strcpy
; Line 57
        PopIP           L00442
; Line 59
L00452:
; Line 60
        LdngA3          $s(00e9)
        CallS2          printf
; Line 61
        LdcW1
        CallS1          exit
; Line 62
        PopIP           L00442
L00441:
        PushIPb         L00442
        CaseW
        BeginSwitch
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   3f
        CaseLabRefShort L00449
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   42
        CaseLabRefShort L00450
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   4b
        CaseLabRefShort L00445
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   4d
        CaseLabRefShort L00447
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   62
        CaseLabRefShort L00450
        CaseEntry       e:0 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   6b
        CaseLabRefShort L00445
        CaseEntry       e:1 w:0 n:0 s:0 l:0 x:0
        CaseValueByte   6d
        CaseLabRefShort L00447
        EndSwitch
        Gotonb          L00452
L00442:
; Line 64
        IncfW0          k
; Line 65
; Line 67
L00437:
        LdngBb          szUnit
        JnzW3           L00454
; Line 68
        LdngA2          $s(00fa)
        LdngA0          szUnit
        CallS0          strcpy
; Line 69
L00454:
        LdcW3
        LdpW0           argvx
        LdfW0           k
        LdinoW
        LdfA3           szDrive
        CallS3          strncpy
; Line 70
        ClrfBb          szDrive+2
; Line 71
        LdpW0           argvx
        LdfW0           k
        LdinoW
        CallSbb         SetDrive
; Line 72
        StfpWb          ret
        LdcW0
        JgeWb           L00457
; Line 73
; Line 74
        LdngA1          $s(0100)
        CallS2          printf
; Line 75
        LdcW1
        CallS1          exit
; Line 76
; Line 77
L00457:
L00433:
        Ret
        EndProc         GetOptions
; chDrive = 0004
; Line 82
        BeginProc       native near cdecl SetDrive
        Chkstk          0000
; Line 84
        LdpW0           chDrive
        LdinB0
        JzrWb           L00462
; Line 85
; Line 87
        LdpW0           chDrive
        LdinB0
        CvtBW
        LdcWub          41
        JltWb           L00464
        LdpW0           chDrive
        LdinB0
        CvtBW
        LdcWub          5a
        JleWb           L00463
L00464:
        LdpW0           chDrive
        LdinB0
        CvtBW
        LdcWub          61
        JltWb           L00465
        LdpW0           chDrive
        LdinB0
        CvtBW
        LdcWub          7a
        JleWb           L00463
L00465:
; Line 88
        LdcWm1
        Ret
; Line 90
L00463:
        LdpW0           chDrive
        LdinB1
        CvtBW
        LdcWub          3a
        JeqWb           L00466
; Line 91
        LdcWm1
        Ret
; Line 93
L00466:
        LdpW0           chDrive
        LdinB2
        JzrW2           L00467
; Line 94
        LdcWm1
        Ret
; Line 95
L00467:
        LdngAb          _ctype
        IncW1
        LdpW0           chDrive
        LdinB0
        CvtBW
        LdinoB
        LdcW2
        AndW
        JnzW4           L00495
        LdpW0           chDrive
        LdinB0
        CvtBW
        Goto5           L00496
L00495:
        LdpW0           chDrive
        LdinB0
        CvtBW
        DecWb           20
L00496:
        DecWb           40
        StngWb          DriveNo
; Line 96
; Line 97
        Goto3           L00468
L00462:
; Line 98
        LdngAb          DriveNo
        CallS0          _dos_getdrive
L00468:
```

```
; Line 99
        LdcW0
; Line 100
L00461:
        Ret
        EndProc         SetDrive
; argc = 0004
; argv = 0006
; ret = fffc
; dtable = fff4
; Line 104
        BeginProc       native near cdecl main
        Chkstk          0000
; Line 108
        LdpW0           argc
        LdcW3
        JleWb           L00477
; Line 109
        CallSb          ShowHelp
; Line 110
L00477:
        LdpW0           argc
        LdcW1
        JleWb           L00478
; Line 111
        LdpW1           argv
        CallSb          GetOptions
; Line 112
        Gotob           L00479
L00478:
; Line 113
; Line 114
        LdngA1          $s(0110)
        LdngAb          szUnit
        CallS5          strcpy
; Line 115
        LdcW0
        CallS4          SetDrive
; Line 116
        StfpWb          ret
        LdcW0
        JgeWb           L00481
; Line 117
; Line 118
        LdngA0          $s(0116)
        CallS3          printf
; Line 119
        LdcW1
        CallS2          exit
; Line 120
; Line 121
L00481:
L00479:
; Line 123
        LdfA3           dtable
        LdngWb          DriveNo
        CallS1          _dos_getdiskfree
; Line 127
        LdcW0
        LdfW0           dtable+6
        LdfW1           dtable+4
        MulW
        LdcW0
        LdfW2           dtable+2
        MulL
        StngLb          freeBytes
; Line 129
        CallS0          ShowFreeBytes
; Line 130
        LdcW0
; Line 131
L00474:
        Ret
        EndProc         main
; Line 136
        BeginProc       native near cdecl ShowFreeBytes
        Chkstk          0000
; Line 137
        LdngAb          $s(0126)
        LdngA1          szUnit
        CallS0          strcmp
        JnzWb           L00484
; Line 138
        LdngLb          freeBytes
        LdcW0
        LdcWw           0400
        DivuL
        LdngAb          $s(012d)
        CallSbb         printf
; Line 139
        Gotob           L00487
L00484:
        LdngAb          $s(0131)
        LdngA1          szUnit
        CallS0          strcmp
        JnzWb           L00488
; Line 140
        LdngLb          freeBytes
        LdcWub          10
        LdcW0
        DivuL
        LdngA4          $s(0138)
        CallSbb         printf
; Line 141
        Gotob           L00491
L00488:
; Line 142
        LdngLb          freeBytes
        LdngA3          $s(013c)
        CallSbb         printf
L00491:
L00487:
; Line 143
        LdngA1          szUnit
        LdngA2          $s(0140)
        CallSbb         printf
; Line 144
L00483:
        Ret
        EndProc         ShowFreeBytes

```

リスト 4　C++のサンプルプログラム

```
#include <stdio.h>
#include <iostream.h>

class Point
{
protected:
        int xx;
        int yy;
public:
        Point() {yy = 0, xx = 0;}
        Point (int x, int y) {xx = x, yy = y;}
        void Show (void);
        Point& operator + (Point&);
        Point& operator = (Point&);
};


void
Point::Show (void)
{
        cout << "xx = " << xx;
        cout << "¥nyy = " << yy << "¥n";
}

Point&
Point::operator+ (Point& aPoint)
{
        Point *ans;

        ans = new Point ();

        ans->xx = aPoint.xx + xx;
        ans->yy = aPoint.yy + yy;
        return *ans;
}

Point&
Point::operator= (Point& aPoint)
{
        Point *ans;

/*  ans = new Point ();

        ans->xx = aPoint.xx;
        ans->yy = aPoint.yy;
*/
        xx = aPoint.xx;
        yy = aPoint.yy;
        return      *ans;
}

main ()
{
        Point a(10, 11), b(5, 6), c;

        a.Show();
        b.Show();

        c = a + b;
        c.Show();
        return 0;
}

```


Windows プログラミング用語集
ダイナミックリンキング（Dynamic linking）　実行ファイル作成時にすべての関数をリンクせず，実行時にアドレスを決定し，残りの関数をリンクすること．

(5) Windows 3.0のサポート

MS-C/C++ 7.0では，デフォルトがWindows 3.1用のプログラム開発になっている．したがって3.1用に作成されたWindowsプログラムは，Windows 3.0のリアルモードでは動作しない．しかし日本を見てもわかるとおり，すべてが3.1に移行したわけではない．MS-C/C++ 7.0では，そのような時のために3.0用のリアルモード対応プログラムを開発するためのオプションが用意されている．秋以降に各社から日本語Windows 3.1が発売されると思われるので，それまでのつなぎに使うオプションと見てもよいだろう．次期バージョンのMS-C/C++ 7.0では削除されるか，3.1用のオプションに変更されるかもしれない．

そのほかにも，さまざまな拡張がMS-C/C++ 7.0には施されている．たとえば，今まであったクイックコンパイル (-qc) に代わってファーストコンパイル (-f) がある．もちろん-qcオプションもそのまま健在だが，こちらはMS-C 6.0との互換性のために残されていて，マニュアルでは-fオプションを使うように推奨している．

またCコンパイラとして見ても，よりANSIの規格に準拠するようになっている．いくつかの手持ちのプログラムをコンパイルしてみたのだが，MS-C 6.0では出なかった警告がMS-C/C++ 7.0では小姑の小言のように湧き出してきた．

その他

その他にもいくつかの拡張が施され，より強力なプログラムが簡単に作成できるようになっている．

MOVEのサポート

MOVE (Microsoft Overlay Virtual Environment) とは，てっとり早くいえばオーバーレイのことである．今までのオーバーレイといえば，やたらと制限があり，デバッグがやりにくく，ディスクからロードされるために実行が遅いといった不満がかなりあった．DOSの640KBの壁を越えるために致し方なく利用していた感があったのだが，最近は，EMSやXMSの利用によってそれらの問題が解決されようとしている．しかしEMS, XMSを利用するためにはそれらの規約を理解しなくてはならず，一般にあまり利用されてはいなかった．MOVEはこれらの問題を解決し，より簡単にオーバーレイプログラムが作成できる機能を提供してくれるのである．

MOVEは，まずオーバーレイプログラムのXMSへの配置を試みる．もしうまくいけばメデタシメデタシであるが，失敗した場合はEMSへの配置を試みてくれる．さらに失敗するようであれば，ディスクへ配置というぐあいに，今までプログラマが自分でやらなければならないことをいくつかの関数で実現してくれるのである．これならばよほどのことがない限りEMS辺りに配置される確立が高くなるから，今までより高速にプログラムを実行できるようになる．

またMOVEは，オブジェクト単位でオーバーレイを指定するのではなく関数単位で指定できるため，より柔軟なプログラムが書けるようになる．オーバーレイの入れ子も許可されているために，複数のオーバーレイモジュールをメモリに配置することも可能になっている．

MOVE.LIBで提供される関数群を表1に示す．

VMのサポート

MOVEがプログラムコードをXMS/EMS/DISKに配置するのに対し，VM (Virtual Memory Library) は，データをXMS/EMS/DISKに配置する．ちょうどalloc系関数の仮想メモリ対応版だと考えることができるであろう．

表1　MOVE.LIBで提供される関数

_moveinit()	_movepause()
_movesetheap()	_moveresume()
_movegetcache()	_movetraceon()
_movesetcache()	_movetraceoff()

XMS/EMS/DISKの，どれに領域を確保するかはプログラマが指定でき，MVにお任せを指定すると，XMS/EMS/DISKの順番で適当な場所を確保してくれる．実メモリの領域を食いつぶさないために，貴重な領域が損なわれなくて非常に助かる．

VMで提供される関数群を表2に示す．

表2 VMで提供される関数

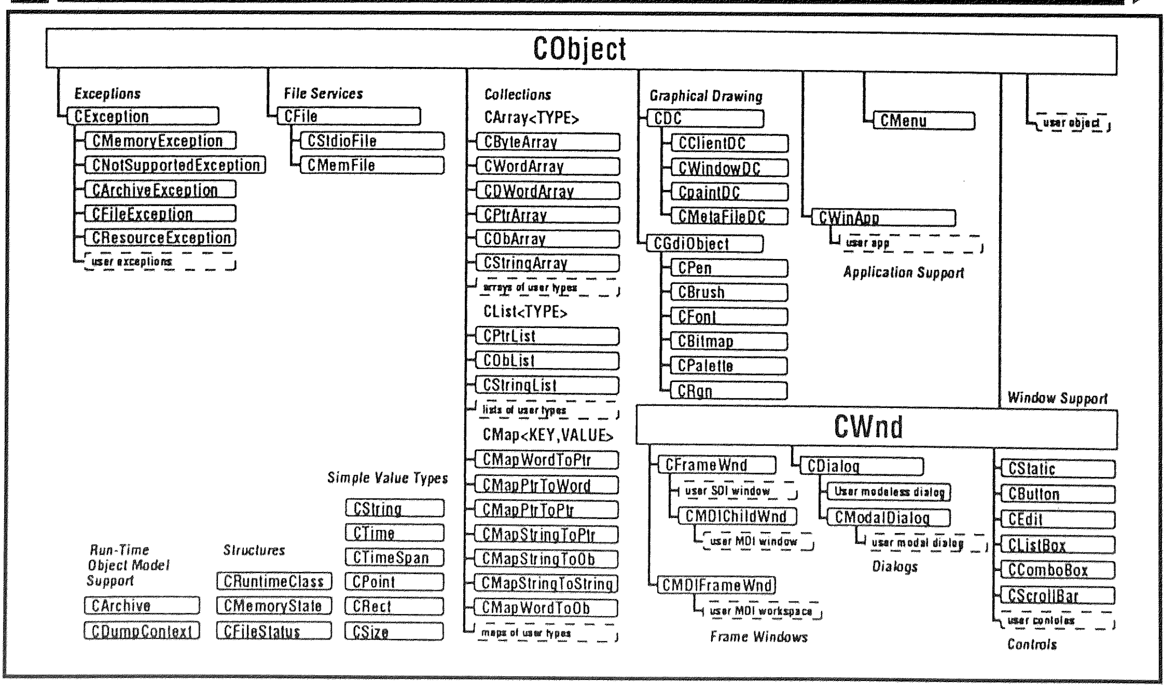
_vfrr()	_vlockcnt()
_vheapinit()	_vmalloc()
_vheapterm()	_vmsize()
_vload()	_vrealloc()
_vlock()	_vunlock()



MS-C/C++ 7.0には，Microsoft Foundation Class Library（以後MFCと略す）と呼ばれるクラスライブラリが付属している．注目すべきは，“MFCは，マイクロソフト社が自社のWindowsのために作成したクラスライブラリ”である点だ．今までの経緯を考えると，このクラスライブラリが，Windowsプログラミングの世界で標準となることは明白であると思われる．参考までに，図4へMFCのClass Hierarchyを掲示する．Smalltalkのクラスライブラリと比べると，まだクラスが不足している感があるのは仕方がないが，最初の製品としてはよくこれだけのものを付属させたと思う．

また実際のプログラミングにおいてはクラスライブラリへ，ユーザーが独自に不足のクラスを追加しなくてはならない場合が発生する．この場合には，図4の”USER OBJECTS”と書かれている部分にユーザーのクラスを追加するように指示されている．そのためユーザーは，どこへクラスを追加すればよいのか迷わなくてすむ．MFCのソースコードはすべて付属しているので，直接ライブラリに追加することも可能なのだが，できるならば参考にする程度にとどめてオリジナルを破壊しないように拡張していくほうが望ましいであろう．マイクロソフト社から拡張クラスライブラリキットなるもの

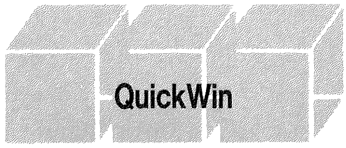
図4 MFCのClass Hierarchy



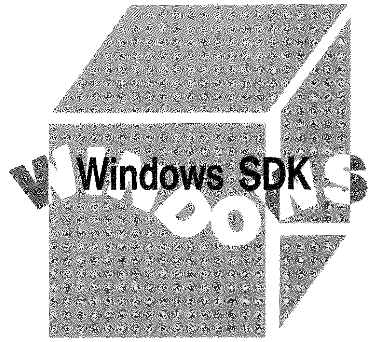
Windowsプログラミング用語集
チェックボックス（Check boxes） ユーザーがオプションを選択する場合などに使われ，ボタンはトグルスイッチになっている．

が出荷されてくれば，さらに嬉しいのだが…….

さて，できあいのクラスを使ってプログラミングを始めると，クラス名とメンバ名がなかなか結び付かないことがある．しかしMFCは，クラス名に"C"+オブジェクト名，メンバ名にAPI名が付いていることが多いので，API名がわかればクラス名とメンバ名が想像できるようになっている．たとえば，TextOutは，Device Contextを利用するからクラスCDCのメンバであると想像することができ，実際にCDC::TextOutとなっている．このようにMFCは，従来のC言語を利用したWindowsプログラマがC++へ移行しやすいように配慮されている（表3）．



Windowsのプログラミングルールを知らなくてもWindows上でのプログラミングができるQuickWinがサポートされている．これは，Qucik C for WindowsやMicrosoft Fortran Version 5.1でもサポートされているので，ご存じの読者も多いだろう．ただし，この機能が利用できるのは，画面やBIOSに直接アクセスしない優良プログラムだけである．



今までに，ざっとMS-C/C++ 7.0

表3　class CDCのメンバ関数

Construction/Destruction
- CDC
- ~CDC

Initiallzation
- CreateDC
- CreateIC
- CreateCompatibleDC
- DeleteDC

Device-Context Function
- GetDCOrg
- SaveDC
- RestoreDC
- GetDeviceCaps

Drawing-Tools Function
- GetBrushOrg
- SetBrushOrg
- EnumObjects

Type-Safe Selection Helpers
- SelectObject
- SelectStockObject

Color and Color Palette Function
- GetNearestColor
- SelectPalette
- UpdateColors

Drawing-Attribute Functions
- GetBkColor
- SetBkColor
- GetBkMode
- SetBkMode
- GetPolyFillMode
- SetPloyFillMode
- GetROP2
- SetROP2
- GetStretchBltMode
- SetStretchBltMode
- GetTextColor
- SetTextColor

Mapping Function
- GetMapMode
- SetMapMode
- GetViewportOrg
- SetViewportOrg
- OffsetViewportOrg
- GetViewportExt
- SetViewportExt
- ScaleViewportExt
- GetWindowOrg
- SetWindowOrg
- OffsetWindowOrg
- GetWindowExt
- SetWindowExt
- ScaleWindowExt

Region Functions
- FillRgn
- FrameRgn
- InvertRgn
- PaintRgn

Clipping Functions
- GetClipBox
- SelectClipRgn
- ExcludeClipRect
- ExcludeUpdateRgn
- IntersectClipRect
- OffsetClipRgn
- PtVisible
- RectVisible

Line-Output Functions
- GetCurrentPosition
- MoveTo
- LineTo
- Arc
- Polyline

Simple Drawing Functions
- FillRect
- FrameRect
- InvertRect
- DrawIcon

Ellipse and Polygon Functions
- Chord
- DrawFocusRect
- Ellipse
- Pie
- Polygon
- PolyPolygon
- Rectangle
- RoundRect

Bitmap Functions
- PatBlt
- BitBlt
- StretchBlt
- GetPixel
- SetPixel
- FloodFill
- ExtFloodFill

Text Functions
- TextOut
- ExtTextOut
- TabbedTextOut
- DrawText
- GetTextExtent
- GetTabbedTextExtent
- GrayString
- GetTextAlign
- SetTextAlign
- GetTextFace
- GetTextMetrics
- SetTextJustification
- GetTextCharacterExtra
- SetTextCharacterExtra

Coordinate Functions
- DPtoLP
- LPtoDP

Font Functions
- FetCharWidth
- SetMapperFlags
- GetAspectRatioFilter

Printer Escape Functions
- Escape
- StartDoc
- StartPage
- EndPage
- SetAbortProc
- AbortDoc
- EndDoc

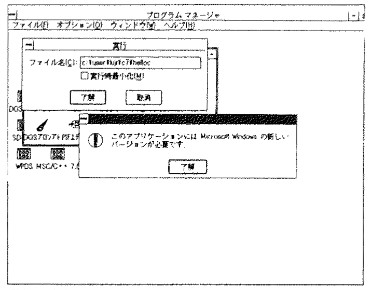
Scrolling Functions
- ScrollDC

MetaFile Functions
- PlayMetaFile

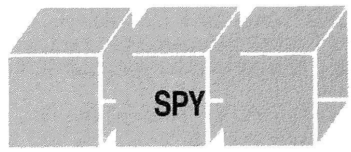
について紹介してきたが，ここからはもう1つの目玉であるWindows SDKについて紹介する．前バージョンまでのMS-C/C++ 7.0には，SDKではなくOS/2のライブラリやツールが含まれていたが，MS-C/C++ 7.0からはSDKそのものが含まれるようになっている．今まで市販されていたSDKには，Windowsのプログラミングを行うために必要な，すべてのノウハウが詰まっていた．そのため非常に膨大なデータが本とディスクになってパッケージ化されていた．MS-C/C++ 7.0には，それがそっくりそのまま梱包されている．ゆえに，前にも述べたがMS-C/C++ 7.0関連の物とあわせるととんでもなく巨大なパッケージになっている．

SDKが対象としているWindowsは，当然バージョン3.1である．しかし日本ではまだ日本語版のリリースが行われていないために，一部の人が3.1を英語環境で動かしているだけで大半の人々は3.0のままになっている．そのためMS-C/C++ 7.0のデフォルト設定で作成したWindowsプログラムは，3.0では動作させることができない（画面15）．しかし，MS-C/C++ 7.0およびSDK（正確にはリソースコンパイラとヘルプコンパイラ）には3.0に対応するためのオプションが（-Gq/-30など）用意されている．3.0対応のプログラムを作成する場合のMAKEFILEを参考に掲載しておく（リスト5）．

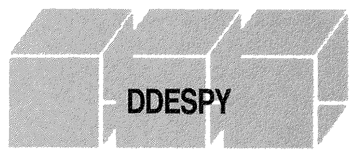
画面4を見てもわかる通り，SDKに入っている開発支援ツールは，アイコンの絵柄が変更されたりQuick C for Windowsで添付されていたツールと入れ換えられたりしている．なお一部のツールは3.0で動作しないか，動作してもヘルプファイルのバージョンが上がっているのでヘルプの表示ができないので注意が必要である（WINHELP.EXEを入れ換えればヘルプは表示できるようになるが，今度は日本語が表示されなくなってしまう）．



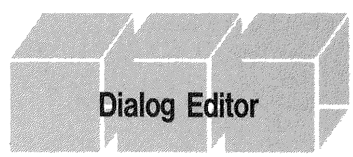
画面15



SPYは，SDK3.0でも付属していたメッセージ監視ツールである．3.0では，アイコンがまるで幽霊のようであったが，3.1では劇画風のスパイアイコンに変わっている．



DDESPYは，DDE通信のメッセージを監視するために新しく追加されたツールである．Windows 3.0では動作せず，システムがハングアップする．



Dialog Editorは，簡単にダイアログを作成するためのツールである．使い勝手その他に変更はない．



リスト 5　Windows 3.0 対応プログラム作成用 MAKEFILE の例

```
# このメイクファイルは，NMAKEでのみ動作します．

PROG      = WinLha
DEBUG     = 0
CFLAGS_S  = -c -AS -W2 -GA -GEf -Zp -DWINVER=0x0300 -Oe -Og -Os -Gs
CFLAGS_D  = -c -AS -W2 -GA -GEf -Zp -DWINVER=0x0300 -f -Od -Zi -Gs

.SUFFIXES:
.SUFFIXES: .res .rc .obj .c .def

all: $(PROG).exe

$(PROG).res: $(PROG).rc $(PROG).ico
    rc -r $(PROG).rc

$(PROG).obj: $(PROG).c $(PROG).h makefile

$(PROG).bin: $(PROG).obj $(PROG).def
!IF $(DEBUG)
    link /NOD /CO $(PROG),$(PROG).bin,, libw slibcew, $(PROG).def
!ELSE
    link /NOD $(PROG),$(PROG).bin,, libw slibcew, $(PROG).def
!ENDIF

$(PROG).exe: $(PROG).bin $(PROG).res
    @copy $(PROG).bin $(PROG).exe >nul
    rc -30 $(PROG).res

.c.obj :
!IF $(DEBUG)
        $(CC) $(CFLAGS_D) $<
!ELSE
        $(CC) $(CFLAGS_S) $<
!ENDIF


```


Windows プログラミング用語集
デバイスコンテキスト（Device context）　GDIによって管理されるデータ構造体で，プリンタや（ディスプレイの）特定のウィンドウと連結している．

Image Editorは，SDKPaintに代わって追加された，アイコン，ビットマップ，カーソル作成ツールである．Quick C for Windowsからバンドルされるようになったツールであるが，Windows 3.0でグリッド表示させるとUAEを発生する．

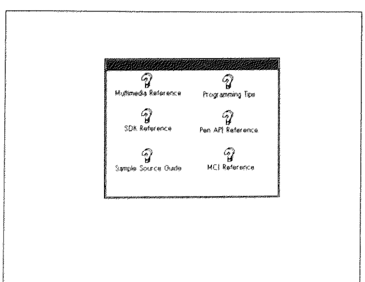
Hot Spot Editor

Hot Spot Editorは，Windows 3.1からサポートされたホットスポットを作成するためのツールである．この機能を利用したヘルプは，新しWindowsヘルプを利用しないと機能しないので注意が必要である．

Stress Application

Stress Applicationは，開発されたWindowsプログラムを過酷な環境でテストするためのツールである．SDK3.0には，SHAKERユーティリティが付属していたが，それに代わる代替えツールである．しかし，SHAKERよりは過酷なテストができるようになっている．Windows 3.0では動作しない．

CodeView Version 3.07



画面16

表4　SDK3.1で提供されるサンプル

blandmdi	helpex	penpad
bouncer	helphook	rainbow
cddemo	hform	reverse
clidemo	hooks	rotary
commdlg	instver	showgdi
cropdib	listhorz	snoop
dbwin	lowpass	srec
ddeml	macrohlp	srvrdemo
defprocs	makeapp	tdosmem
dibit	mcitest	timers
dibview	memory	toolhelp
dragdrop	midimon	tty
expense	muscroll	ver
fontest	mypal	verstamp
gdosmem	ownerb	winmem32
gwapi	palette	wmfdcode
handler	pencntl	xtension

CodeView Version 3.07は，MS-C/C++ 7.0が利用できずに従来のMS-C 6.0を利用するプログラマのために付属しているデバッガである．バージョンが0.7上がっている．

Zoom Utility

Zoom Utilityは，SDK3.0にも付属していた画面拡大ツールである．16色までしか対応していない．

Font Editor

Font Editorは，SDK3.0にも付属していたフォント作成ツールである．Windows 3.0では動作しない．

Heap Waker

Heap Wakerは，SDK3.0にも付属していたメモリ監視ツールである．このツールもWindows 3.0では動作しない．

DBWin Sample Application

DBWin Sample Applicationは，デバッグカーネルを用いているときにデバッグログを記録するためのツールである．このツールもWindows 3.0では動作しない．

その他

従来のSDKと比べてSDK3.1はかなり提供物が増えている．MS-C 6.0用のライブラリもそうだが，Windows for Pen用のアプリケーションを開発するためのドライバやDLL，SETUPプログラムの作り方，マルチメディアなどいたれりつくせりである．

サンプルやオンラインヘルプも増え，これからWindowsプログラミングを始める人たちでも入門しやすいように考慮されている（表4，画面16）．

当然のことながら，SDK3.1は基本

Windowsプログラミング用語集
トラッキング（Tracking）　マウスをドラッグしたときの，ウィンドウやアイコンの動作のこと．トラッキングが遅いと動きがぎこちなくなる．

的にWindows 3.1の上で利用することを前提にしている．DOS/Vユーザーもしくは PC-DOSユーザー以外は，日本語版 Windows 3.1が出るまであまり利用できないだろう．

プロファイラ

プロファイラもMS-C/C++ 7.0に新しくバンドリングされたツールである．これもSDKと同じく単体で販売されていたのとまったく同じものが付属している．

プロファイラは開発ツールというより，プログラム自己批判ツールとでもいったほうが適切なのかもしれない．自作したプログラムをこのツールにかけると，こと細かに批判してくれる．批判といってもけなすのではなく，この関数は何回呼ばれた．とか，この関数の実行には何秒かかった．などと解析結果を通知してくれるのだ．プログラマはその結果を受けて，より速いプログラムになるよう，プログラムに手を加えることになる．

参考までにプロファイラが出力する解析結果の一例を図5に掲載しておく．

```
Microsoft PLIST Version 1.20

Profile: Function counting, sorted by counts.
Date:    Mon Jun 01 16:30:35 1992


Program Statistics
------------------
    Total functions: 6
    Total hits: 558
    Function coverage:  83.3%

Module Statistics for c:¥msc¥profiler¥samples¥c¥dos¥qsort.exe
--------------------------------------------------------------
    Functions in module: 6
    Hits in module: 558
    Module function coverage:  83.3%

 Hit
count    %   Function
---------------------
  360  64.5 SwapWords (qsort.c:183)
  195  34.9 QuickSort (qsort.c:121)
    1   0.2 main (qsort.c:35)
    1   0.2 GetWords (qsort.c:59)
    1   0.2 OutWords (qsort.c:193)
    0   0.0 usage (qsort.c:217)
```

図5 プロファイラの解析結果

マニュアル

MS-C/C++ 7.0には，膨大な量のマニュアルが含まれている．最後にこのマニュアル群の紹介を行う．ただし英語版MS-C/C++ 7.0の場合であって日本語版に同じ量のマニュアルが付属するかどうかは不明である．まったく同じ構成で日本語化されたら，たぶん英語版より量は増えるのではないかと思われる．

386MAX

User's Guide

386MAXのインストール方法やメモリ管理などについて解説している．約44ページ．

Refernce Guide

オプションや付属のツールの解説，システムのチューニング方法を解説している．約136ページ．

C/C++

Getting Started

付属ドキュメントの紹介と，インストールについて解説している．インストールを行う前に1度は目を通しておく必要がある．37ページ．

Comprehensive Index and Error Reference

すべての総索引とエラーメッセージの解説が書かれている．マニュアルが多いので，この索引があると「どのマニュアルの何ページ」とわかるので非常に助かる．558ページ．

C++ Tutorial

C++の入門書．豊富なサンプルを使っていて，難しい英語も使っていないので1度読んでみることをお勧めする．205ページ．

C++ Language Reference

C++の言語仕様とその例が解説されている．前記の本と合わせて読むと

Windowsプログラミング用語集
バイナリラスタオペレーション（Binary raster operation）　ラスタオペレーションの中でも，とくに2つのピクセルパターンしかない論理操作のこと．

よい．452ページ．

C Language Reference

ご存じC言語の言語仕様解説書である．非常にうまくまとめられているが，せっかくC++がサポートされているのだから，読者もこの機会にC++へ移行してもらいたい．291ページ．

Environment and Tools

非常にぶ厚く，PWBやCVなど付属ツールの解説が行われている．874ページ．

Class Libraries User's Guide

MFCを使ってプログラミングを行う方法を豊富なサンプルとともに解説している．C++ Tutorialを読み終わったら，ステップアップを兼ねて読んでおきたい．426ページ．

Class Libraries Reference

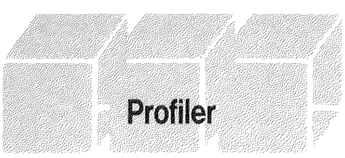
MFCの全クラスライブラリ関数の解説．前出のClass Libraries User's Guideと合わせて読むとよい．1021ページ．

Run-Time Library Reference

今まで通り標準関数の解説．ただし，QuickWinなどの関数解説まで入っているので，若干引きづらくなってしまっている．923ページ．

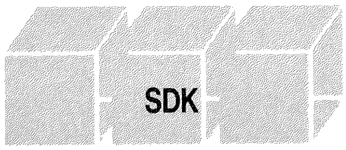
Programming Techniques

P-Codeや他言語との連結方法，最適化などプログラミングやコンパイルオプションの組み合わせなどプログラミングテクニックについて解説している．323ページ．



User's Guide

プロファイラの取り扱い解説書．PWBからの利用方法などを丁寧に解説している．プログラムのパフォーマンスを気にしないプログラマには必要がないが，気にするプログラマには出力結果などの見方も書いてあるので必読．113ページ．



今までのマニュアル紹介で，ゲッソリしている読者もいるだろう．しかしまだ半分しか紹介していない．今度はSDK関連のマニュアル紹介に突入する．ちなみにすべてのマニュアルには，For the Microsoft Windows Operating Systemと書かれている点にも注目したい．

Getting Started

インストールについての解説と，何が新しくなったのかを紹介している．インストールを行う前に1度は目を通しておく必要がある．42ページ．

Setup Toolkit for Windows

今までは，Windowsプログラムのインストレーションのための SETUPファイルをどのように作成したらよいのか一般に公開されていなかった．しかしWindows 3.1からは，コモンダイアログが追加されたために，SETUPファイルの作成のしかた，SETUP.INFファイルのスクリプトの作成方法が本書で細かく解説されている．130ページ．

Programming Tools

開発支援ツールの取り扱い説明書になっている．おもにリソースコンパイラ，ヘルプコンパイラ，デバッガなどのツール解説と，デバッグのためのツール利用方法が書かれている．265ページ．

Programmer's Reference, Volume1. Overview

Windowsプログラミングを行うために必要な解説が，サンプルリストと共に紹介されている．Windowsプログラムを始める前に熟読しておく必要があるであろう．従来のプログラマーズリファレンスと比べて，より独学しやすくなっている．

特筆すべきは，今まで一般に公開されていなかった，プログラムマネージャインターフェースや，圧縮ファイル解凍インターフェースなどもここで紹介されている点である．またWindows 3.1のコントロールパネルに追加されたスクリーンセーバーの作成方法も公開されている．すべてサンプルソース付き．501ページ．

Programmer's Reference, Volume2.Functions

WindowsAPIの解説書である．新しいAPIには3.1と表示されているので，作成した結果旧バージョンのWindowsで動作しないなどの事故がおきにくくなっている（もっともちゃんと注意していればだが……）．またAPIによってはサンプルも一緒に掲載され，今までより読みやすく仕上がっている．1001ページ．

Programmer's Reference, Volume3.Messages, Structures, and Macros

Windowsプログラムを組むにはメッセージの詳細がわかっていなくてはならない．Programmer's Refer-

ence, Volume3.は，メッセージの解説，プログラミングに必要な構造体の解説，#defineマクロなどの解説が行われている．605ページ．

Programmer's Reference, Volume4.Resource

フォントやダイアログなどのリソースファイルの作成方法を解説している．もっとも最近は，他社からリソースを簡単に作成するツールも販売されているので，読まなくてもリソースは作成できるのではないだろうか？　340ページ．

Multimedia Programmer's Guide

マルチメディア（MPC）対応アプリケーションを作成するための入門書になっている．単なる入門書ではなく，MPCプログラミングのすべてがここで解説されているので，そちらの方向にすすみたいプログラマは必読．この時ほどMPCキットを導入していなかったことが悔やまれたことはない．このマニュアルを読んで，今回の筆者の原稿料はMPCキット代に消えることが決定してしまった．約240ページ．

Multimedia Programmer's Reference

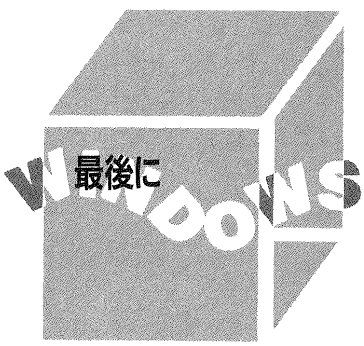
マルチメディアAPI，メッセージのすべてがここにある．MPCはこんなことまでできるのかと感心して指をくわえている．323ページ．

Microsoft Windows for Pen Computing:Programmer's Reference

やっと姿を表しつつあるPen Windows. このマニュアルには，それらに対応するための解説や，追加APIの解説が行われている．316ページ．

The Win32 Application Programming Interface: An Overview

来たるべきWindows NTに備えて，32ビットプログラミングを行うためのガイドラインが紹介されている小冊子．44ページ．



これまで新しいMS-C/C++ 7.0について延々と紹介してきたが，これがMS-C/C++ 7.0のすべてではないことを最後につけ加えておきたい．なぜなら筆者の元にこのパッケージが届いて約3週間，英文マニュアルと格闘しながら利用して，使い方がわかっただけを紹介しているからだ．まだ十分使いこなしていないので，まちがった使い方，知らないツールもあるだろう．この製品はマイクロソフト社最大の製品（中身も箱も）といっても過言ではないと筆者は思っているので，日本語版が楽しみである．

最後に簡単なWindows用プログラムを従来通りC言語を利用して作成した場合とC++言語とMFCを利用した場合のリストを掲載しておく（リスト6～16)．C++とClass Libraryがある場合，プログラミングステップがどれほど短縮されるかを見てもらいたい．プログラミングステップが圧縮されるということは，開発工数も圧縮できると筆者は考えているのだが，読者の感想はいかがだろう．

リスト 6　makefile

```
# このメイクファイルは，NMAKEでのみ動作します.

PROG       = helloc
DEBUG      = 1
CFLAGS_S   = -c -AS -W2 -GA -GEf -Zp -DWINVER=0x0300 -Oe -Og -Os -Gs
CFLAGS_D   = -c -AS -W2 -GA -GEf -Zp -DWINVER=0x0300 -f -Od -Zi -Gs

.SUFFIXES:
.SUFFIXES: .res .rc .obj .c .def

all: $(PROG).exe

$(PROG).res: $(PROG).rc TP.ico
    rc -r $(PROG).rc

$(PROG).obj: $(PROG).c $(PROG).h makefile

$(PROG).bin: $(PROG).obj $(PROG).def
!IF $(DEBUG)
    link /NOD /CO $(PROG).$(PROG).bin,, libw slibcew, $(PROG).def
!ELSE
    link /NOD $(PROG).$(PROG).bin,, libw slibcew, $(PROG).def
!ENDIF

$(PROG).exe: $(PROG).bin $(PROG).res
    @copy $(PROG).bin $(PROG).exe >nul
    rc -30 $(PROG).res

.c.obj :
!IF $(DEBUG)
        $(CC) $(CFLAGS_D) $<
!ELSE
        $(CC) $(CFLAGS_S) $<
!ENDIF
```

リスト 7　helloc.def

```
NAME        Cooper      ; アプリケーションのモジュール名
```


Windows プログラミング用語集
ハンガリー記法（Hungarian notation）　短い小文字のデータ型名を変数名の先頭に付記し，変数名からデータ型を判断できるようにした記名法

```
DESCRIPTION          'MS-WINDOWS Application'
EXETYPE                  WINDOWS
STUB                     'WINSTUB.EXE'
CODE  PRELOAD MOVEABLE DISCARDABLE
DATA  PRELOAD MOVEABLE MULTIPLE
HEAPSIZE                 1024
STACKSIZE   8192
EXPORTS
        MainWndProc   @1   ; ウィンドウ処理関数の名前.
        About          @2   ; "About"処理関数の名前.
```

リスト 8 helloc.c

```
#include <windows.h>

#include "helloc.h"

HANDLE hInst;
#define TITLE       (LPSTR)"The OWls are not what they seem."

int PASCAL
WinMain (HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine,int nCmdShow)
{
    MSG msg;

    if (!hPrevInstance)
        if (!InitApplication (hInstance))
            return FALSE;

    if (!InitInstance (hInstance, nCmdShow))
        return FALSE;

    while (GetMessage (&msg, NULL, NULL, NULL))
    {
        TranslateMessage (&msg);   /* 実際のキー コードへの変換 */
        DispatchMessage (&msg);      /* ウィンドウへのメッセージ送信 */
    }
    return msg.wParam;
}


BOOL
InitApplication (HANDLE hInstance)
{
    WNDCLASS    wc;

    wc.style = NULL;
    wc.lpfnWndProc = MainWndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(hInstance, "TPIcon");
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = GetStockObject (WHITE_BRUSH);
    wc.lpszMenuName =  "TPMenu";
    wc.lpszClassName = "TPWClass"; /* CreateWindow によって使う名前 */

    return RegisterClass (&wc);

}


BOOL
InitInstance (HANDLE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance;

    hWnd = CreateWindow (
        "TPWClass",
        "Twin Peaks",
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT,          /* デフォルト水平位置 */
        CW_USEDEFAULT,          /* デフォルト垂直位置 */
        CW_USEDEFAULT,          /* デフォルト幅       */
        CW_USEDEFAULT,          /* デフォルト高さ     */
        NULL,
        NULL,
        hInstance,
        NULL);

    if (!hWnd)
        return FALSE;

    ShowWindow (hWnd, nCmdShow);
    UpdateWindow (hWnd);
    return TRUE;
}


long FAR PASCAL
MainWndProc (HWND hWnd, unsigned message, WORD wParam, LONG lParam)
{
    FARPROC lpProcAbout;    /* "About" 関数へのポインタ   */
    PAINTSTRUCT ps;
    HDC         hDC;
    RECT    rcWindow;

    switch (message)
    {
        case WM_COMMAND:
            if (wParam == IDM_ABOUT)
            {
                lpProcAbout = MakeProcInstance (About, hInst);
                DialogBox (hInst, "AboutBox", hWnd, lpProcAbout);
                FreeProcInstance (lpProcAbout);
            }
            else
                return DefWindowProc (hWnd, message, wParam, lParam);
            break;

        case WM_PAINT:
            hDC = BeginPaint (hWnd, &ps);
            GetWindowRect (hWnd, (LPRECT)&rcWindow);
            DrawText (hDC, TITLE, -1, &rcWindow, (DT_CENTER | DT_VCENTER));
            EndPaint (hWnd, &ps);
            break;


        case WM_DESTROY:
            PostQuitMessage (FALSE);
            break;

        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return NULL;
}


BOOL FAR PASCAL
About (HWND hDlg, unsigned message, WORD wParam, LONG lParam)
{
    switch (message)
    {
        case WM_INITDIALOG:
            return TRUE;

        case WM_COMMAND:
        if (wParam == IDOK || wParam == IDCANCEL)
        {
            EndDialog (hDlg, TRUE);
            return TRUE;
        }
        break;
    }
    return FALSE;
}
```

リスト 9 helloc.h

```
#define IDM_ABOUT 100
#define IDM_LAURA 101

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE, int);
long FAR PASCAL MainWndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);
```

リスト 10 helloc.rc

```
#include <windows.h>
#include "helloc.h"

TPIcon ICON tp.ico

TPMenu MENU
BEGIN
  POPUP        "ヘルプ(¥036H¥037^)"
  BEGIN
    MENUITEM "Dale Cooperについて(¥036D¥037=)...", IDM_ABOUT
    MENUITEM "Laura Palmerについて(¥036L¥037=)...", IDM_LAURA
  END
END

AboutBox DIALOG 22, 17, 144, 75
CAPTION "Dale Cooperについて"
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
BEGIN
        CTEXT "FBI", -1, 0, 5, 144, 8
        CTEXT "Twin Peaks", -1, 0, 14, 144, 8
        CTEXT "Version 3.0", -1, 1, 33, 144, 8
        CONTROL "了解", IDOK, "BUTTON", WS_GROUP, 80, 53, 32, 14
        ICON "TPIcon", -1, 23, 48, 16, 14, WS_CHILD | WS_VISIBLE
END
```

リスト 11　makefile.cpp

```
CPPFLAGS=  /DWINVER=0x0300 /AS /W3 /Zp /GA /GEs /G2
LINKFLAGS=/NOD /ONERROR:NOEXE

!if "$(DEBUG)"=="1"
CPPFLAGS=/D_DEBUG $(CPPFLAGS) /Od /Zi /f
LINKFLAGS=$(LINKFLAGS) /COD
LIBS=safxcwd libw slibcew
!else
CPPFLAGS=$(CPPFLAGS) /Oselg /Gs
LINKFLAGS=$(LINKFLAGS)
LIBS=safxcw libw slibcew
!endif

hello.exe:     hello.obj hello.def hello.res
    link $(LINKFLAGS) hello, hello, NUL, $(LIBS),hello.def;
    rc -30 /t hello.res


hello.res:  resource.h tp.ico hello.rc

hello.obj:  hello.h resource.h

clean:
    -erase hello.exe
    -erase hello.res
    -erase hello.obj

```

リスト 12　hello.rc

```
#include <windows.h>
#include <afxres.h>
#include "resource.h"

AFX_IDI_STD_FRAME    ICON    tp.ico

MainMenu MENU
{
        POPUP        "&ヘルプ"
        {
                MENUITEM "&Dale Cooperについて...¥tF1", IDM_ABOUT
            MENUITEM "&Laura Palmerについて...¥tF2", IDM_LAURA
        }
}

MainAccelTable ACCELERATORS
{
        VK_F1,       IDM_ABOUT,  VIRTKEY
        VK_F2,       IDM_LAURA,  VIRTKEY
}

AboutBox DIALOG 24, 33, 144, 75
CAPTION "Dale Cooperについて"
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
BEGIN
        CTEXT "FBI", -1, 0, 5, 144, 8
        CTEXT "Twin Peaks", -1, 0, 14, 144, 8
        CTEXT "Version 3.0", -1, 1, 33, 144, 8
        CONTROL "了解", IDOK, "BUTTON", WS_GROUP, 80, 53, 32, 14
        ICON AFX_IDI_STD_FRAME, -1, 23, 48, 16, 14, WS_CHILD | WS_VISIBLE
END



```

リスト 13　hello.cpp

```
#include <afxwin.h>
#include "resource.h"

#include "hello.h"

CTheApp theApp;

CMainWindow::CMainWindow()
{
    LoadAccelTable( "MainAccelTable" );
    Create( NULL, "Twin Peaks/C++ Version",
        WS_OVERLAPPEDWINDOW, rectDefault, NULL, "MainMenu" );
}

void CMainWindow::OnPaint()
{
    CString s = "The OWls are not what they seem.";
    CPaintDC dc( this );
    CRect rect;

    GetClientRect( rect );
    dc.SetTextAlign( TA_BASELINE | TA_CENTER );
    dc.SetTextColor( ::GetSysColor( COLOR_WINDOWTEXT ) );
    dc.SetBkMode(TRANSPARENT);
    dc.TextOut( ( rect.right / 2 ), ( rect.bottom / 2 ),
                s, s.GetLength() );
}

void CMainWindow::OnAbout()
{
    CModalDialog about( "AboutBox", this );
    about.DoModal();
}

BEGIN_MESSAGE_MAP( CMainWindow, CFrameWnd )
    ON_WM_PAINT()
    ON_COMMAND( IDM_ABOUT, OnAbout )
END_MESSAGE_MAP()

BOOL CTheApp::InitInstance()
{
    TRACE( "The OWls are not what they seem.¥n" );

    m_pMainWnd = new CMainWindow();
    m_pMainWnd->ShowWindow( m_nCmdShow );
    m_pMainWnd->UpdateWindow();

    return TRUE;
}

```

リスト 14　hello.h

```
#ifndef __HELLO_H__
#define __HELLO_H__

class CMainWindow : public CFrameWnd
{
public:
    CMainWindow();

    afx_msg void OnPaint();
    afx_msg void OnAbout();

    DECLARE_MESSAGE_MAP()
};

class CTheApp : public CWinApp
{
public:
    BOOL InitInstance();
};

#endif // __HELLO_H__

```

リスト 15　hello.def

```
NAME          Cooper2

DESCRIPTION   'MFC MS-Windows Application'
EXETYPE       WINDOWS
STUB          'WINSTUB.EXE'
CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE MULTIPLE
HEAPSIZE      1024
STACKSIZE     4096


```

リスト 16　resource.h

```
#define IDM_ABOUT    100
#define IDM_LAURA    101

```


Windows プログラミング用語集
ハンドル (Handle)　ハンドルは16ビットのunsigned int型の値で，Windowsが内部参照するために使う数値

ObjectWindowsを使ったWindowsプログラミング

Borland C++ & Application Frameworks 3.0

大野元久

Borland C++には，Windowsアプリケーションをより簡単に作成するためのフレームワークとして，ObjectWindowsと呼ばれるクラスライブラリが付属している．本項では，ObjectWindowsの活用と実践的なプログラミングノウハウを紹介する．

はじめに

Borland C++ 2.0が登場して1年半，日本語化が行われてからも1年がたちました．当初，数多くの要望があったユーザーインターフェースを構築するためのクラスライブラリについても，Application Frameworksとして Windows用のObjectWindowsとDOS用のTurboVisionがサポートされ，C++による開発環境がひととおり揃いました．

最新版であるBorland C++ 3.0では，ボーランドの言語製品ではじめてグローバルオプティマイザが採用されています．また，DPMIとVCPIのサポートやWindowsホストの統合環境など，Windows/DOSアプリケーション開発の環境は一層充実したといえるでしょう．

しかし，マニュアル以外にはこれといった解説書がないため，プログラミングにとまどいを覚えている方も少なくないようです．ここでは，Borland C++ 3.0におけるプログラミング環境の設定からObjectWindowsによるプログラミングまでを概説します．

必要な環境

Borland C++ 3.0でプログラミングする際に必要なものは，メモリ，ハードディスク，CPUパワーです．

コンパイラは，すでにMS-DOSの640KBのメモリ空間だけでは処理しきれない規模になっています．コマンドラインコンパイラであっても，最低1MBのプロテクトメモリが必要です．つねにプロテクトモードで動作するため，許容できるプログラムサイズも十分に大きなものです．ただし，ソースファイルの大きさに関わらず1つの関数が1000行以上もあるような大きなものであると内部のワークエリアがオーバーフローしてしまうようです．

DOS上の統合環境（BC）を使うのであれば，最低2MBくらいのプロテクトメモリは欲しいところです．とくに，Windowsから呼び出すことを考えれば5～8MBくらいのプロテクトメモリを増設したほうがよいでしょう．メモリは，拡張スロットで拡張するタイプのものではいけません．ぜひCPUボードに搭載できるものを使ってください．日本ではEMSが使えればよいというアプリケーションが多いので拡張スロットに増設するEMS/XMS両用のタイプのものが広く使われていますが，このタイプのメモリのアクセス速度はCPUがV30の頃のものとあまり変わりません．Borland C++ 3.0ではプロテクトメモリをメインメモリとして使いますから，処理速度が非常に遅くなってしまいます．

ハードディスクは，Borland C++をインストールするためだけでも50MB必要です．もちろん，アプリケーション開発のための作業エリアが別に必要です．プリコンパイルヘッダ機能は数100KB～数MBの空き領域を必要としますし，ライブラリを構築しようとしても数MBの空き領域が必要

Windowsプログラミング用語集
ヒープ（Heap）　サイズが可変の変数領域．ダイナミックアロケーションされたローカルメモリオブジェクトを保持するのに使用される．

となります．必ずしもすべてが必要なわけではありませんが，Borland C++ 2.0がすでにインストールされていて，それを削除して使うのでなければ1台くらいは増設するつもりでいたほうがよいかもしれません．空き容量が不足しているときでもライブラリソースコードはできるだけインストールするようにしてください．プロフェッショナルの方でもライブラリソースコードをお持ちでない方が多いようですが，ライブラリ関数の動作を調べるのはライブラリソースコードを見るのが最も確実な方法だからです．

CPUパワーについても，Windowsが快適に使える程度のものは必要です．動作環境としては286でも使えるのですが，386以上でないと使いづらいかもしれません．ただ，Borland C++ 3.0は2.0のころに比べるとコンパイル速度が向上しているので，2.0のBCX/BCCXよりは快適に使えるでしょう．プロテクトメモリを拡張スロットで増設している場合は，どんなに速いCPUを使っていても処理速度は遅いままです．

プログラミング環境を構築する

Windows 3.0がインストールされていなければ，まずWindowsをインストールし，それからBorland C++ 3.0をインストールしてください．Borland C++をインストールした後でWindowsを起動すると，自動的にBorland C++のグループを作成してくれます．

さて，起動ドライブのCONFIG.SYSには自動的にいろいろなドライバが組み込まれていると思いますが，PC-9801ではどのデバイスドライバを使うかということは重要な問題です．たとえば，EMM386はEMSを使う多くのアプリケーションのためには便利なものですが，そのままではBorland C++ 3.0と併用できません．

DOSのようにOS自体がプロテクトモードの管理をしていない場合，プロテクトモードで動作するプログラムは自分で管理をしなければなりませんが，こうしたプログラムは同時には1つしか動作できません．ほかのプログラムがプロテクトモードを使おうとしても，元のプログラムがそれを「プロテクト」してしまうのです．

Borland C++ 3.0では，プロテクトモードを複数のアプリケーションで共用するためにVCPIやDPMIというインターフェースをサポートしています．しかし，これらのインターフェースは相手側も同じインターフェースをサポートしていなければなりません．残念なことに，NEC製のDOSやWindowsに付属しているEMM386，MEMORY-PRO386などPC-9801用の仮想86EMSドライバと呼ばれるものの多くはこれらをサポートしていません．

幸いなことに，メルコ製のMelware 5がVCPIに対応しています．EPSON機をお使いの方は，EPSON製のDOS 5.0に付属しているEMM 386がVCPIに対応しています(注：DOS 5.0に付属しているEMM386でVCPIに対応していないのはNEC製のものだけです)．これらの上では，Windowsをエンハンスドモードで起動することもできますから，Borland C++の開発環境にはぴったりです．Windows 3.0のエンハンスドモードから呼び出すDOSプロンプトではDPMIがサポートされていますから，ここでもBorland C++を使うことができます．エンハンスドモードのDOSプロンプトに限れば，EMM386が組み込まれていてもだいじょうぶです．

筆者は，PC-9801RAでMELEMM.386を常用しています．MELEMM.386でも，指定するオプションによってはBCとの相性が悪くなることもあるようですが，VJE-βやVzを組み込むこともできますし，そこからBCを起動したりWindowsのエンハンスドモードを起動してそこからBCを呼び出したりできます．もちろん，WindowsホストのTurbo C++ for Windowsを使えばこうしたことを考えずにすむのですが，Windows環境下のFEP(かな漢字変換フロントプロセッサ)を使い慣れないことと，すでにDOS用に作成したツールが多いのでキャラクタベースの統合環境を使うことのほうが多いのです．

Windowsプログラミングのおさらい

ページ数が限られていますから，ここではWindowsプログラミングのうちメッセージによるプログラミングの概略だけを説明します．

プログラムが，すべてメッセージを通じて行われるということはWindowsプログラミングの中で最も重要なことです．メッセージによる処理とはどういうものか，古典的なプログラミングと比較してみましょう．たとえば，ボタンが押されて文字列が表示されるという処理に対してリスト1のようなプログラムが考えられます．

メッセージを使ったプログラミングではリスト2のようになります．

かえって複雑なプログラミングになっているようにも見えますが，プログラムの拡張性を考えるとメッセージによるプログラミングのほうが役に立つ

Windowsプログラミング用語集
ピクセル（Pixels）　画素を意味し，画面表示の最小情報単位．

のです．たとえば，古典的なスタイルでは入力処理と出力処理が切り放せないものとなっているため，リターンキーを押した場合にも文字列を表示したいという場合，入力と出力の部分を書き換えることになります．メッセージによるプログラミングでは，リスト3のcase文を追加するだけですみます．

メッセージによるプログラミングは，プログラムの個々の処理を独立させるために役立ちます．処理が独立することで，ある部分の処理を変更しても，それがほかの部分に影響することは少なくなります．これは，信頼性の高いアプリケーションを開発するためには重要なことです．

もうひとつ重要なことは，処理の単位をメッセージで区切り，入力待ちや無駄なループをなくすことでマルチタスクを実現しやすくしていることです．

Windowsは，DOSというシングルタスクOSの上でマルチタスクを実現しています．DOS自体はリエントラント（再入可能）でないため，ハード的な割り込みによってほかのプログラムに処理を移すことはできません．DOSの処理中に割り込みがかかったら，DOS自身が破壊されてしまうおそれがあるからです．逆に割り込みをかけなければ，プログラムのある処理にシステムが独占されてしまうと，ほかのプログラムが動作できなくなります．

Windowsでは，メッセージを使うことで個々の処理にシステムを独占させないようにしています．Windowsは起動されたアプリケーションにメッセージを分配することで，割り込みに頼らずに複数のアプリケーションを実行させています．このため，あるアプリケーションがメッセージ処理に時間がかかってしまうと，ほかのアプリケーションはその間は何もできなくなってしまいます．

実際のWindowsでは，メッセージは次の4つの部分から成り立ちます．

- **ウィンドウハンドル**
- **メッセージの種類**
- **WORDパラメータ**
- **LONGパラメータ**

ウィンドウハンドルによって，どのウィンドウに対するメッセージかを識別し，WM_xxxであらわされるメッセージの種類によってその目的を分類します．WORDパラメータとLONGパラメータは，メッセージの種類に対するもっと細かい情報を提供します．情報が，WORDやLONGで収まりきらない場合は，LONGパラメータは適

リスト 1

```
while  (終了条件でない)  {
    ボタンが押されたかどうかのチェック;
    if  (ボタンが押された)  {
        文字列の表示ルーチンを呼び出す.
    }
}
```

リスト 2

```
while  (終了条件でない)  {
    メッセージの受取り;
    switch  (メッセージの種類)  {
    case  マウスの処理:
        if  (マウスボタンが押された)
            「ボタンが押された」メッセージの生成;
        break;
    case  ボタンが押された:
        「文字列の表示」メッセージの生成;
        break;
    case  文字列の表示:
        実際に文字列を表示する;
        break;
    default:
        「マウスの処理」メッセージの生成;
        「キーの処理」メッセージの生成;
        break;
    }
}
```

リスト 3

```
case  キーの処理:
    if (リターンが押された)
        「文字列の表示」メッセージの生成;
    break;
```

リスト 4

```
while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

当な構造体型へのポインタを指すことになります．

また，Windowsプログラムには必ずメッセージループと呼ばれる繰り返し文が存在します．これは，リスト4のようなものです．

GetMessageがメッセージを受け取り，DispatchMessageがメッセージに従って適当なウィンドウ関数に対してメッセージを渡します．TranslateMessageは，キー入力メッセージを変換するために使われ，アクセラレータという機能を使う場合はこの部分を変更しなければなりません．

Windows上のプログラミングは，ウィンドウをオープンするために必ずウィンドウクラスというものをWindows自身に登録しなければなりません．ウィンドウクラスは，ウィンドウの基本的な性質を構造体（WNDCLASS）で表現したものです．この性質には，ウィンドウが実際にどのような動作をするかということも含まれます．もちろん，Cでは構造体の中に直接関数を記述することはできませんから，動作をあらわす関数へのポインタが含まれているだけです．

ウィンドウの動作はウィンドウ関数（window procedure）と呼ばれるものであらわされます．ウィンドウ関数は，基本的には前述したメッセージを分岐させるためのswitch文からなります．受け取ったメッセージをどう処理するかがそのウィンドウクラスの性質を決定するわけです．

C++のおさらい

C＋＋プログラミングについても，ざっと解説します．

C＋＋の特徴は，ごく簡単に説明するならば構造体を拡張したものということができるでしょう．Cにおける構造体は，単一のデータ型だけでは表現できないものを複数のデータ型をまとめることで表現するために使われます．たとえば，住所録を考えましょう．単に名前を表現するための文字列だけでは何の役にも立ちませんから，名前と住所とをまとめて1つのデータ型とするために構造体が使われます．これで，名前と住所とを関連づけて処理することができます（リスト5）．

しかし，一般にプログラムはデータだけからなるものではありません．あるデータがあれば，必ずそれに対する処理が存在するはずです．Cにおける構造体はデータ型だけしか含むことができませんので，それを処理する関数は，ほかの関数と同様グローバルに存在することになります（リスト6）．

C＋＋ではその構造体にだけ作用する関数を構造体のメンバとして記述することができます．つまり，データだけでなくその挙動という性質も構造体で表現できます．C＋＋では，このような構造体をクラスと呼びます．先に述べたWindowsのためのクラスにもウィンドウの動作をあらわすために関数へのポインタが用意されていました．Cの構造体はデータメンバしか定義できないため，このようになっていたのです．C＋＋では，関数そのものを構造体の中に入れてしまうことができます．

クラスの中に定義した関数をメンバ関数と呼びます．リスト7の例では，person，setname，setaddr，printがメンバ関数です．また，personのようにクラスと同じ名前を持つメンバ関数をコンストラクタと呼びます．コンストラクタは，そのクラスを一般的には初期化するために使われ，そのクラス型のインスタンス（変数）が定義されるたびに暗黙的に呼び出されます．もし，コンストラクタがなければ，引数を持たずに何もしないデフォルトコン

リスト 5

```
struct person {
  char name[20];  // 名前
  char addr[50];  // 住所
};
```

※//（2つのスラッシュ）は，C++で拡張されたコメントの形式です．

リスト 6

```
void print_person(struct person man)
{
  // 名前と住所の表示
  puts(man.name);
  puts(man.addr);
}
```

リスト 7

```
struct person {
  char name[20];
  char addr[50];
  person() { name[0] = '¥0'; addr[0] = '¥0'; }
  void setname(const char *s)
  { strcpy(name, s); }
  void setaddr(const char *s)
  { strcpy(addr, s); }
  // 名前と住所の表示
  virtual void print() {
     puts(name);
     puts(addr);
  }
};
```


Windowsプログラミング用語集
ビューポイント（View point）　マッピングは物理座標で行われ，たいていの場合ビューポイントはクライアントエリアと一致する．

リスト 8

```
struct newperson: person {
  char tel[20]; // 電話番号
  newperson() { tel[0] = '¥0'; }
  void settel(const char *s)
  { strcpy(tel, s); }
  void print() {
    // 元々のデータを表示
    person::print();
    // 追加したデータを表示
    puts(tel);
  }
};
```

リスト 9

```
class person {
// class のデフォルトは private
// クラスの外部からはアクセスできない
  char name[20];
  char addr[50];
public:
  person() { name[0] = '¥0'; addr[0] = '¥0'; }
  void setname(const char *s)
  { strcpy(name, s); }
  void setaddr(const char *s)
  { strcpy(addr, s); }
  virtual void print() {
     puts(name);
     puts(addr);
  }
};
```

ストラクタと呼ばれるものが暗黙的に生成されます.

メンバ関数printの前についているvirtualという予約語は仮想関数を定義するためのもので，次で述べる機能のために必要なものです.

C++のクラスは，その性質を拡張するための機能を備えています．リスト7の例で人 (person) に対する情報として電話番号を追加したいとします．Cでは，構造体personそのものを変更しなければなりません．これは，プログラムが拡張されるたびにすでにデバッグした元のプログラムを変更しなければならないことを意味します．小さなプログラムではさほど問題になりませんが，大きなプログラムを拡張していくたびに元のプログラムまでデバッグし直さなければならないとしたらデバッグの負担が重くなり，信頼性を保つのが難しくなります.

C++では，継承という機能を使います．リスト8の例をみてください.

ここでは，新しい構造体newpersonを定義するのに':'を使って元になるクラスを指定しています．元になるクラスを基本クラス(base class)，拡張したクラスを派生クラス (derived class) と呼びます．派生されたクラスでは，基本クラスのデータやメンバ関数が使えるようになります．しかし，コンストラクタだけは継承されません．基本クラスに引数を持つコンストラクタがあっても，派生クラスではそのコンストラクタの形式をそのまま使うことはできないのです.

newpersonでは，telというデータメンバとsettelというメンバ関数が追加されています．また，personの中にあったprintというメンバ関数もあらわれています．クラスを拡張しようとするときは，単にデータメンバやメンバ関数を追加するだけでなく，元のクラスのメンバ関数そのものを拡張しなければならないことがあります．このとき元のクラスにvirtualがついていなくても，そのメンバ関数を定義することができるのですが，そうすると基本クラスへのポインタを使った場合に正しいメンバ関数が呼び出されないことがあるのです．クラスを設計するときに，それを継承するクラスが機能を変更・拡張しなければならないだろうメンバ関数にはvirtualをつけておくべきです.

また，newperson::printでは基本クラスのメンバ関数person::printを明示的に呼び出しています．基本クラスのメンバ関数は，派生クラスのメンバ関数から暗黙的に呼び出されることはありません．コンストラクタが基本クラスから派生クラスの順に呼び出されてインスタンスを初期化するので勘違いすることがあるようですが，コンストラクタはあくまでもそのクラスの初期化に必要なので呼び出されているのです．実際には，クラスの定義にはstructの代わりにclassという予約語を使います．また，基本的にデータメンバはprivate:という領域で定義し，外部からクラスにアクセスするときはメンバ関数を経由して行うようにします（リスト9).

この継承という考え方が，ObjectWindowsでプログラムする際に非常に重要になってきます．ObjectWindowsでは，ウィンドウやアプリケーションのメッセージ処理を行う基本的なクラスがあらかじめ定義されています．プログラマは，そのクラスから新たなクラスを派生することでそのクラスの性質を拡張できます．ObjectWindowsで提供されているクラスは，単なるウィンドウだけではありません．エディットウィンドウやファイルを編集するためのウィンドウなど，よく使われそうな高い機能を持つクラスも提供されていますから，これらを使えば簡単に高機能なプログラムを開発できるわけです.

Windowsプログラミング用語集
フォーカス（Focus)　複数のウィンドウの中で，キーボードからのメッセージを受け取る権利のあるウィンドウをフォーカスのあるウィンドウという.

Windowsプログラムを開発するために

BCを用いてWindowsプログラムを作成するときには，いったんWindowsを起動しておいてそこからBCを呼び出すほうがよいでしょう．自動的に作成されたBorland C++グループのBC3アイコンにはBC.PIFが割り当てられています．WindowsからBC.EXEを呼び出すことはできませんので，必ず提供されたBC.PIFを使うようにしてください．Windowsから呼び出されたBCからはGRPH+TABを押して，すぐWindowsに切り換えたりWindowsから制御を移すことができます．Windowsのエンハンスドモードでは，この切り換えにはほとんど時間はかかりませんから，ためしに実行してみたりデバッグする際にはとても便利です．

必ずしも統合環境（BC）を使う必要はありませんが，エディタとコマンドラインを組み合わせて開発していると，処理を切り換えるのがうっとおしくなるものです．慣れたエディタが使いたい場合でも，BCのトランスファ機能を使えばそれを呼び出せます．BC自体は，DOSのメモリからは70KB程度しか消費しませんからメモリ不足になることもないでしょう．BCのプロジェクト機能は，リソースファイル（.RES）でも，モジュール定義ファイル（.DEF）でも，何でもプロジェクトに追加するだけで自動的に処理を振り分けてくれますから，これを使わない手はありません．

だから，BCを使うときは必ずプロジェクトをオープンします．Windowsアプリケーションを開発するためには，Options | Applicationダイアログボックスで Windows Appを選択します．これで，コード生成など最低限必要なオプションが指定されます．コード生成の効率に配慮するならばOptions | Compiler | Entry/Exit CodeではWindows explicit functions exportedを選択します．ObjectWindowsを使うときは，Windows smart callbacksを指定します．また，これ以外にOptions | Linker | LibrariesでContainer Class LibraryとObjectWindows LibraryをNoneからStaticに変更しておきます．

もちろん，ディスクの空き領域に余裕さえあればOptions | Compiler | Code generation | Pre-compiled headersをチェックしておくべきです．ObjectWindowsを使おうとすると，最低でも8000行近いヘッダファイルをコンパイルしなければなりませんから，どんなにコンパイルが速くてもそれなりに時間がかかってしまいます．プリコンパイルヘッダ機能を使えば，この8000行を一度だけコンパイルしておき，後でその情報を再利用できるため開発時間が随分短縮できるはずです．なお，ヘッダファイルが追加されたり順序が変更されるとプリコンパイル情報が再利用できなくなります．しばしばヘッダファイルの内容が変わるときは，最低限必要なowl.hだけにプリコンパイルを適用するため#include <owl.h>の次の行に#pragma hdrstopを入れておくとよいでしょう．

インクルードファイルやライブラリファイルのディレクトリが指定されていなければ，Options | Directoriesに必要なディレクトリを追加します．たとえば，C:¥BORLANDCにデフォルトでインストールしている場合はInclude DirectoriesをC:¥BORLANDC¥INCLUDE;C:¥BORLANDC¥OWL¥INCLUDE;C:¥BORLANDC¥CLASSLIB¥INCLUDEとし，Library DirectoriesをC:¥BORLANDC¥LIB;C:¥BORLANDC¥OWL¥LIB;C:¥BORLANDC¥CLASSLIB¥LIBとします．

ObjectWindows用のモジュール定義ファイルOWL.DEFがOWL¥LIBに用意されていますが，これは必ずしもプロジェクトに追加する必要はありません．

これらは，基本的にはTurbo C++ for Windows（TCW）を使う場合でも同じです．ただし，TCWにはトランスファ機能がないので，ほかのエディタを呼び出すことはできません．また，DOSコマンドのRC（Resource Compiler）を呼び出してリソーススクリプト（.RC）をリソースファイル（.RES）にコンパイルすることはできません．リソースは，Resource Workshopを使って統括的に編集できますからさほど問題ではないでしょう．ただ，これ以降に示すリソースはリソーススクリプト（.RC）形式になっていますから，TCWを使う場合はあらかじめバイナリリソースにコンパイルしておかなければなりません．

コマンドラインコンパイラを使う場合は，やはりTURBOC.CFGにインクルードディレクトリ（-I）とライブラリディレクトリ（-L）の指定にCLASSLIBとOWLの内容を追加しておいてください．コマンドラインオプションには-W（-WS）を指定し，プログラム名とともにTCLASSS.LIBとOWLWS.LIBを指定してください．OWLのサンプルプロジェクトをPRJ2MAKでメイクファイルに変換してその内容を見てみるのもよいかもしれません．

Windowsプログラミング用語集
ブラシ（Brushes） ビットマップ化されたピクセルパターンを定義するGDI関数のひとつ．閉じた領域の塗りつぶしなどに利用される．

プログラムマネージャのBorland C++グループには開発中のプログラムとデバッグ用のアイコンを追加しておくことをお勧めします．よほど小さいプログラムでなければ，プログラムの開発中は何度もプログラムやデバッガを起動することになるでしょう．デバッグ用のアイコンは，あらかじめ用意されているTDWアイコンとは別にそのプログラムをデバッグするためのものを追加します．ファイル|新規でコマンドラインに「TDW -t<開発中のデイレクトリ>　<プログラム名>」を指定します．たとえば，C:¥WORK¥HELLOでHELLOプログラムを開発しているのであれば，TDW　-tC:¥WORK¥HELLO HELLOとします．

リソースの作成

リソースを作成するためには，Resource Workshopを使うのが最適です．Resource Workshopは，すべてのリソースを視覚的に編集できますし，またプロジェクトとして統括的に管理できるため，リソースのためのファイルが分割されずにすみます．Resource Workshopを使えばリソーススクリプトで記述されたリソースはもちろん，バイナリリソース(.RES)や実行ファイル(.EXE，.DLL)に組み込まれたリソースを編集したり，取り出して別のリソースとして活用することができます．既存のツールとの使用感の違いによる慣れの問題はあるかもしれませんが，現在最も機能的なリソースエディタといえるでしょう．

詳しい使用法の解説はマニュアルに譲るとして，ここではリソースをプロジェクトとして作成する場合の手順について述べます．

Resource Workshopを起動したら，リソースを作成するためにFile|New Projectで.RCファイルを選択してください．そのアプリケーションに必要なものが，アイコンだけだったり，ビットマップだけの場合でも，できるだけ.RCを選ぶようにしてください．新しいプロジェクトにはuntitled.rcという名前がつけられますから，とりあえずFile|Save file asでアプリケーションの名前でプロジェクトを保存しておきましょう．必要ならば，ディレクトリパスも指定してください．

次にリソースのIDを定義するためのヘッダファイルをプロジェクトに追加します．File|Add file to projectで適当なヘッダファイル名を指定すれば，このファイルがプロジェクトに追加されます．こうすることで，リソースを編集して新たなIDを決めるたびにその内容がヘッダファイルに反映されるようになります．自分自身でわざわざ識別子の対応表をつくらなくてすむというわけです．ヘッダファイル名は，アプリケーションを構築するほかのものと重複しない名前にしてください．Resource Workshopはコメントを残しませんので，リソース以外の定義と併用しようとするとResource Workshopが勝手にコメントを削除してしまうことがあるからです．

リソースは，バイナリリソースとしてプロジェクトに組み込むほうが効率がよくなりますから，File|Preferencesで Multi-saveの.RESチェックボックスをチェックしてください．ここで，リソースファイル名が正しいものであることも確認します．バイナリリソースが作成されていないと，BCのプロジェクト機能はRC(Resource Compiler)を呼び出してリソーススクリプトをコンパイルしなければなりません．TCWでは，RCを呼び出すことができないためにプロジェクトの構築ができなくなってしまいます．

準備ができたら，後はResource|NewでResource typeリストボックスから適当なものを選び編集するだけです．Resource Workshopには，さまざまな機能が用意されていますからぜひ使い込んでみてください．

なお，この解説で使用したリソースファイルもResource Workshopを使って作成したものです．ただし，Resource Workshopは#include <windows.h>を生成しないのでこの行だけは後で追加しました．また，若干見栄えをよくするためにスペースの調整を行っています．

C++とメッセージ処理

ObjectWindowsを使う前に，C++でのWindowsメッセージ処理について解説します．

リスト10，11，12は，基本的なWindowsのためのクラスを定義したものです(本誌'91年11月号に掲載したものとほぼ同じ)．このクラス定義では派生したクラスでウィンドウ関数を定義するよう，基本クラスでWndProcを純粋基本関数として定義しています．しかし，この方法では，各クラスのウィンドウ関数の中にはおきまりのswitch文と数多くのcase文を用意しなければならなくなります．

Windowsにおけるメッセージはそれぞれ独立したものですから，メッセージごとに別々のメンバ関数が呼び出されるようにしたほうがよいでしょう．このためには，リスト13のようなクラ

リスト10 win.h

```
// win.h:
//      Windows アプリケーションのためのクラスを宣言したヘッダファイル

#if !defined(__cplusplus)
#error Must specify C++ option.
#endif

#if !defined(__WIN_H)
#define __WIN_H

#if !defined(__WINDOWS_H)
#include <windows.h>
#endif

#if !defined(__STDLIB_H)
#include <stdlib.h>          // for the function 'exit'
#endif

//
//  アプリケーション名の宣言
//      実体をどこかで定義しなければならない。
//
extern char szAppName[];

//
//  この Windows アプリケーション全体に共通する
//  変数や関数のためのクラス
//
class Main {
public:
    static HANDLE hInstance;        // インスタンス
    static HANDLE hPrevInstance;    // 以前に起動したインスタンス
    static LPSTR lpszCmdLine;       // コマンドライン引数
    static int nCmdShow;            // 表示状態
    static int MessageLoop(void);   // メッセージループ
};

//
//  ウィンドウのための基本抽象クラス
//      すべてのウィンドウは、このクラスから派生する。
//      各派生クラスはそれぞれ WndProc を用意しなければならない。
//
class Window {
public:
    HWND hWnd;      // ウィンドウハンドル
    // ウィンドウハンドルを返す
    HWND GetHandle(void) { return hWnd; }
    // ウィンドウを表示する
    BOOL Show(int nCmdShow) { return ShowWindow(hWnd, nCmdShow); }
    // ウィンドウの内容を更新する
    void Update(void) { UpdateWindow(hWnd); }
    // ウィンドウごとのウィンドウ関数
    virtual long WndProc(WORD message, WORD wParam, LONG lParam) = 0;
};

//
//  ウィンドウクラスのためのクラス
//      ウィンドウクラスを登録するためには、このアプリケーションに
//      適したウィンドウクラスを指定して Register() を呼び出す
//      派生クラスを用意しなければならない。
//      このクラスは、windows.h の WNDCLASS 構造体から派生している。
//

// このアプリケーションのためのウィンドウ関数のプロトタイプ宣言
long FAR PASCAL _export WndProc(HWND, WORD, WORD, LONG);

class WinClass : private WNDCLASS {
    BOOL bRegistered;   // ウィンドウクラスの登録状態
public:
    // デフォルトのウィンドウクラスの設定
    WinClass(void) {
        style = CS_HREDRAW | CS_VREDRAW;
        lpfnWndProc = ::WndProc;
        cbClsExtra = 0;
        cbWndExtra = sizeof (Window *);
        hInstance = Main::hInstance;
        hIcon = LoadIcon(NULL, IDI_APPLICATION);
        hCursor = LoadCursor(NULL, IDC_ARROW);
        hbrBackground = GetStockObject(WHITE_BRUSH);
        lpszMenuName = NULL;
        lpszClassName = szAppName;

        bRegistered = FALSE;
    }
    // ウィンドウクラスのパラメータを変更するためのメンバ関数
    void SetClassStyle(WORD newstyle) { style = newstyle; }
    void SetClassWndProc(long (FAR PASCAL *lpfnNewProc)(HWND, WORD,
            WORD, LONG)) { lpfnWndProc = lpfnNewProc; }
    void SetClassClsExtra(int cbNewClsExtra) { cbClsExtra = cbNewClsExtra; }
    void SetClassWndExtra(int cbNewWndExtra) { cbWndExtra = cbNewWndExtra; }
    void SetClassIcon(HICON hNewIcon) { hIcon = hNewIcon; }
    void SetClassCursor(HCURSOR hNewCursor) { hCursor = hNewCursor; }
    void SetClassBackground(HBRUSH hbrNewBackground)
        { hbrBackground = hbrNewBackground; }
    void SetClassMenuName(LPSTR lpszNewMenu) { lpszMenuName = lpszNewMenu; }
    void SetClassName(LPSTR lpszNewName) { lpszClassName = lpszNewName; }
    LPSTR GetClassName(void) { return lpszClassName; }
    // ウィンドウクラスを登録する
    void Register(void) {
        if (!RegisterClass(this))
            exit(FALSE);
        bRegistered = TRUE;
    }
    // ウィンドウクラスの登録状態を返す
    BOOL isRegistered(void) { return bRegistered; }
};

#pragma warn -aus

#endif  /* __WIN_H */
```

リスト11 win.cpp

```
#include "win.h"

//
//  Main クラスの本体
//      この Windows アプリケーション全体に共通する変数や関数
//
HANDLE Main::hInstance = 0;             // インスタンス
HANDLE Main::hPrevInstance = 0;         // 以前に起動したインスタンス
LPSTR Main::lpszCmdLine = NULL;         // コマンドライン引数
int Main::nCmdShow = 0;                 // 表示状態

//
//  メッセージループ
//
int Main::MessageLoop(void)
{
    MSG msg;

    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}

//
//  Window クラスから派生したクラスのウィンドウ関数を呼び出す
//  ディスパッチ用のウィンドウ関数の定義。
//

//
//  ウィンドウエクストラからクラスへのポインタを取得したり
//  取得するためのインライン関数
//
inline void *GetPointer(HWND hWnd)
{
#if defined(__SMALL__) || defined(__MEDIUM__)
    return (void *)GetWindowWord(hWnd, 0);
#elif defined(__LARGE__) || defined(__COMPACT__)
    return (void *)GetWindowLong(hWnd, 0);
#else
#error use Small, Medium, Compact or Large model
#endif
}

inline void SetPointer(HWND hWnd, void *ptr)
{
#if defined(__SMALL__) || defined(__MEDIUM__)
    SetWindowWord(hWnd, 0, (WORD)ptr);
#elif defined(__LARGE__) || defined(__COMPACT__)
    SetWindowLong(hWnd, 0, (WORD)ptr);
#else
#error use Small, Medium, Compact or Large model
#endif
}

//
//  ウィンドウ関数の本体
//
long FAR PASCAL _export WndProc(HWND hWnd, WORD message, WORD wParam,
                                LONG lParam)
{
    // ウィンドウエクストラから Window オブジェクトへのポインタを取得する。
    Window *pWin = (Window *)GetPointer(hWnd);

    if (pWin == NULL) {
        if (message == WM_CREATE) {
            LPCREATESTRUCT lpcs;

            // ウィンドウエクストラに Window オブジェクトを
            // 設定する。これによってウインドウエクストラから
            // Window オブジェクトのメンバ関数をアクセスできる。
            lpcs = (LPCREATESTRUCT)lParam;
            pWin = (Window *)lpcs->lpCreateParams;
            SetPointer(hWnd, pWin);
            pWin->hWnd = hWnd;
            return pWin->WndProc(message, wParam, lParam);
        }
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return pWin->WndProc(message, wParam, lParam);
}
```

スを定義することができます．こうしておけば，派生クラスでは必要なメッセージに対するメンバ関数を用意することができます．これでメッセージごとの独立性も確保でき，プログラムの可読性も向上します（リスト14）．

これで，めでたしめでたしとならないのがC++のやっかいなところです．

派生クラスが，すべてのメッセージに対するメンバ関数を用意するとは限

Windows プログラミング用語集

プレフィクス（Prefix (notation)）　ハンガリー記法のように変数名の前につけるデータ型名などを指す．

リスト 12 hellowin.cpp

```
//
// HelloWin アプリケーション
//
#include "win.h"
#include <string.h>

//
// アプリケーション名の定義
//
char szAppName[] = "HelloWin";

//
// アプリケーション固有のクラスの定義
//

//
// HelloWin アプリケーションのためのウィンドウクラスを登録するクラス
//
class WinApp : public WinClass {
public:
    WinApp() { Register(); }
};

//
// HelloWin アプリケーションが使うウィンドウのためのクラス
//
class HelloWindow : public Window {
    char *szHelloMessage;
public:
    HelloWindow(char *szHello) {
        szHelloMessage = szHello;
        hWnd = CreateWindow(szAppName, szAppName, WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, 0,
            CW_USEDEFAULT, 0,
            NULL, NULL, Main::hInstance, (LPSTR)this);
        Show(Main::nCmdShow);
        Update();
    }
    long WndProc(unsigned iMessage, WORD wParam, LONG lParam);
};

//
// HelloWindow のメインウィンドウのためのウィンドウ関数
//
long HelloWindow::WndProc(unsigned iMessage, WORD wParam, LONG lParam)
{
    switch (iMessage) {
    case WM_PAINT:
        PAINTSTRUCT ps;

        BeginPaint(hWnd, &ps);
        TextOut(ps.hdc, 0, 0, szHelloMessage, strlen(szHelloMessage));
        EndPaint(hWnd, &ps);
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hWnd, iMessage, wParam, lParam);
    }
    return 0L;
}

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine,
                    int nCmdShow)
{
    // アプリケーション全体に共通する変数の定義
    Main::hInstance = hInstance;
    Main::hPrevInstance = hPrevInstance;
    Main::lpszCmdLine = lpszCmdLine;
    Main::nCmdShow = nCmdShow;

    // アプリケーション固有の部分
    // ウィンドウクラスの登録
    WinApp MyWin;
    // メインウィンドウ
    HelloWindow HelloWin("Hello, Windows!");

    // メッセージループの呼び出し
    return Main::MessageLoop();
}
```

リスト 13 newwin.h

```
         ......

class Window {
protected:
    HWND hWnd;        // ウィンドウハンドル
public:
    HWND GetHandle(void) { return hWnd; }
    BOOL Show(int nCmdShow) { return ShowWindow(hWnd, nCmdShow); }
    void Update(void) { UpdateWindow(hWnd); }
    virtual long Null(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
    virtual long Create(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
    virtual long Destroy(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
    virtual long Move(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
    virtual long Size(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
    virtual long Paint(WORD message, WORD wParam, LONG lParam)
    { return DefWindowProc(hWnd, message, wParam, lParam); }
         ......

    virtual long WndProc(WORD message, WORD wParam, LONG lParam)
    {
        switch (message) {
        case WM_NULL: return Null(message, wParam, lParam);
        case WM_CREATE: return Create(message, wParam, lParam);
        case WM_DESTROY: return Destroy(message, wParam, lparam);
        case WM_MOVE: return Move(message, wParam, lParam);
        case WM_SIZE: return Size(message, wParam, lParam);
        case WM_PAINT: return Paint(message, wParam, lParam);
            ......
        }
    }
};
         ......
```

リスト 14 newhello.cpp

```
※説明を簡単にするため，実装部をすべてクラス内で記述しています．

         ......

class HelloWindow : public Window {
    char *szHelloMessage;
public:
    HelloWindow(char *szHello);
    // WM_PAINT の処理
    long Paint(WORD message, WORD wParam, LONG lParam)
    {
        PAINTSTRUCT ps;
        BeginPaint(hWnd, &ps);
        TextOut(ps.hdc, 0, 0, szHelloMessage, strlen(szHelloMessage));
        EndPaint(hWnd, &ps);
        return 0;
    }
    // WM_DESTROY の処理
    long Destroy(WORD message, WORD wParam, LONG lParam)
    {
        PostQuitMessage(0);
        return 0;
    }
};
         ......
```

りません．むしろ，そのようなケースはほとんどないでしょう．このため，基本クラスの各メンバ関数は純粋仮想関数でなくデフォルトの処理を持った関数として定義されていなければなりません．リスト13で，すべてのメッセージ処理関数がDefWndProcを呼び出すように定義されているのはこのためです．これが仮想関数でなければ単なるインライン関数として定義されるので実際にはコードは生成されないのですが，仮想関数は必ず関数のアドレスを持たなければならないので実体が生成されてしまいます．C++では，異なる関数のアドレスを同じにする方法はありませんので，DefWndProcを呼び出すだけの関数がメッセージの数だけ用意されることになります．これは，どう考えても無駄なことです．

もっとやっかいなことに，仮想関数は通常のメンバ関数とは違い間接的な呼び出しに対応するためのテーブルを持っています．リスト14の派生クラスではただ2つの仮想関数だけを定義していますが，C++はその構造上，継承したすべての仮想関数に対するテーブルを作成してしまいます．Windows

の基本的なメッセージは100種類以上ありますから，これらのメッセージに対応した仮想関数を用意すると，基本クラスやそこから派生したクラスを定義するたびに200～400バイトのデータ領域が消費されることになります．また，メニューコマンドなども同じように処理しようとすると，消費されるデータ量はもっと増えます．普通は，派生クラスで処理したいメッセージの数はそれほど多くありませんから，ほとんど内容の変わらない仮想関数テーブルがたくさんできてしまうことになり，仮想関数テーブルのためにクラスごとに無駄な領域が消費されてしまうことになるのです．

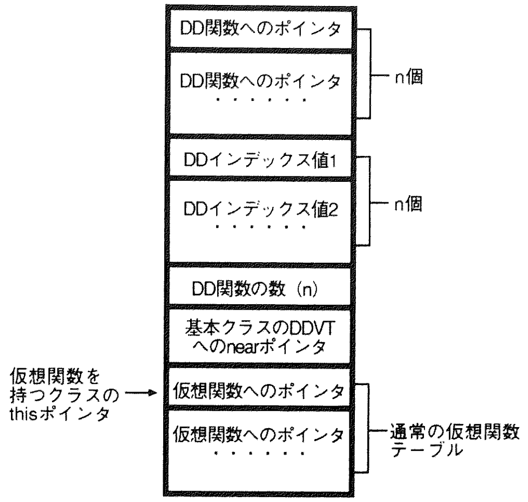


図1

細かいことをいえば，将来新たに処理すべきメッセージが追加されるとメッセージを分岐しているウィンドウ関数そのものも継承して拡張しなければならないことになります．できるだけ拡張性の高いプログラミングを目指したいのですから，メッセージが追加されたときに追加すべきなのは対応するメンバ関数だけにしたいものです．

こういう無駄をなくしたり拡張性を保つためのうまい手はないものでしょうか．C++だけではうまい手がないので，Borland C++では仮想関数を数値で指定して呼び出せるように拡張されています．これがダイナミックディスパッチ（DD）機構と呼ばれるものです．ダイナミックディスパッチ機構を使うと，そのクラスで定義された仮想関数と指定された数値に対応するテーブル（ダイナミックディスパッチバーチャルテーブル，DDVT）を自動的に生成してくれます．これによって，C++の厳密性を保ちながら効率を悪

リスト 15　ddwin_.h

```
※説明を簡単にするため，実装部をすべてクラス内で記述しています.

            ......

typedef long (Window::*_mfunc_t)(WORD, LONG);
typedef long (far *_func_t)(WORD, LONG);

class Window {
protected:
    HWND hWnd;      // ウィンドウハンドル
public:
    HWND GetHandle(void) { return hWnd; }
    BOOL Show(int nCmdShow) { return ShowWindow(hWnd, nCmdShow); }
    void Update(void) { UpdateWindow(hWnd); }
    virtual long WndProc(WORD message, WORD wParam, LONG lParam)
    {
        // DDVT の処理
        union {
            _mfunc_t mfptr;
            _func_t fptr;
        } temp;
        int i, num, *ndx;
        void near * near *curp = *(void near * near **)this;

        temp.mfptr = NULL;
        while (curp) {
            curp--;
            num = *((int *)curp - 1);
            ndx = ((int *)curp - 1 - num);
            for (i = 0; i < num; i++)
                if (ndx[i] == message) {
                    temp.fptr = *((_func_t *)((int *)curp -1 -num) -num +i);
                    goto skip;
                }
            curp = (void near * near *)*curp;
        }
    skip::
        if (temp.mfptr)
            return (this->*temp.mfptr)(wParam, lParam);
        else
            return DefWindowProc(hWnd, message, wParam, lParam);
};
            ......
```

リスト 16　ddhello_.cpp

```
※説明を簡単にするため，実装部をすべてクラス内で記述しています.

            ......

class HelloWindow : public Window {
    char *szHelloMessage;
public:
    HelloWindow(char *szHello) {
        szHelloMessage = szHello;
        hWnd = CreateWindow(szAppName, szAppName, WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, 0,
            CW_USEDEFAULT, 0,
            NULL, NULL, Main::hInstance, (LPSTR)this);
        Show(Main::nCmdShow);
        Update();
    }
    // WM_PAINT の処理
    long Paint(WORD wParam, LONG lParam) = [WM_PAINT]
    {
        PAINTSTRUCT ps;
        BeginPaint(hWnd, &ps);
        TextOut(ps.hdc, 0, 0, szHelloMessage, strlen(szHelloMessage));
        EndPaint(hWnd, &ps);
        return 0;
    }
    // WM_DESTROY の処理
    long Destroy(WORD wParam, LONG lParam) = [WM_DESTROY]
    {
        PostQuitMessage(0);
        return 0;
    }
};
            ......
```


Windows プログラミング用語集
プロシージャ（Procedure）　手続きとか手順を指す，値を返す関数（Function）に対して使われ，ブロックやサブルーチンと同じ．

化させずにメッセージをきれいに処理できるのです.

DDVTの形式を図1に示します.

リスト13, 14をダイナミックディスパッチ機構を使って改良した例をリスト15, 16に示します. リスト15に示されているとおり, Windowクラスの中で実際にディスパッチ（分岐）処理を行っているメンバ関数WndProcは, DDVTの内部構造に依存する面倒な処理を行っています. しかし, ObjectWindowsを使う限りこの部分はObjectWindows自身が処理してくれますからプログラマが気にする必要はありません.

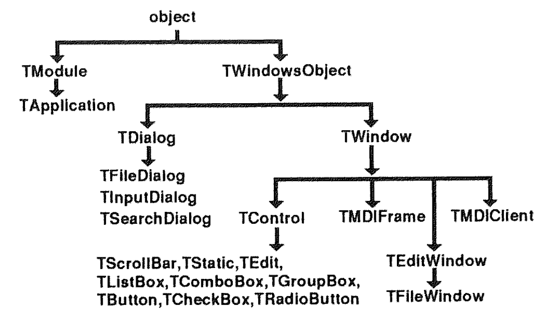


図2

ObjectWindows プログラミングの基礎

まず, ObjectWindowsの基本的なクラス構成を図2に示します.

ObjectWindowsのプログラミングは, C++継承を使って拡張する形で行うことが基本です. ObjectWindowsで, 最も簡単なプログラム例をリスト17に示します. このプログラム自身は, 1つもクラスを派生させていません. 唯一使われているTApplicationクラスは, ObjectWindowsにあらかじめ用意されているものです. では, TApplicationはどんなことをしているのでしょうか.

まず, TApplication型のインスタンス（変数）Smallを宣言するときにアプリケーションの名前とWinMainに渡された4つの引数を指定します. アプリケーションの名前は, おもにメッセージボックスを表示するときに使われます. 引数は, それぞれこのアプリケーションのインスタンスハンドル, 最初に起動されたインスタンスハンドル, コマンドライン, 表示指定を表します. これらは, TApplicationやその基本クラスが解析します.

TApplicationは, ウィンドウのための基本クラスTWindowをメインウィンドウとしてオープンしています. デフォルトでは, TWindowにはTApplicationの名前がキャプションとしてそのまま渡されるため, リスト17はアプリケーションの名前として指定された文字列“SmallApp”をそのままウィンドウのキャプション（タイトル）として使います.

さて, TApplicationではInitMainWindowという仮想関数を呼び出してメインウィンドウをオープンしています. これはリスト18のような内容になっています.

MainWindowはTApplication内のデータメンバで, アプリケーションのメインウィンドウを表します. TWindowは, 一般的なウィンドウの核となるクラスで引数には親ウィンドウのハンドル（親がないときはNULL）とウィンドウのタイトルを持ちます. TWindow自身は, ほとんどなにもしません. つまり, なにもしないウィンドウをアプリケーションのメインウィンドウにしているわけです.

ここで, TApplicationとTWindowの関係について述べておきます. TWindowは, TWindowsObjectという基本クラスから派生したものですが, TWindowsObjectからはTDialogという重要なクラスも派生しています. また, ボタンやリストボックスなどのコントロールはTWindowから派生したものです. 視覚的な

リスト 17 small.cpp

```
#include <owl.h>
int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    TApplication Small("SmallApp", hInst, hPrevInst, lpCmd, nShow);
    SmallApp.Run();
    return SmallApp.Status;
}
```

リスト 18

```
void TApplication::InitMainWindow()
{
  MainWindow = new TWindow(NULL, Name);
}
```

リスト19 hello.cpp

```
#include <owl.h>

class THelloApp: public TApplication {
public:
    // コンストラクタ
    THelloApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    // InitMainWindow のオーバーライド
    void InitMainWindow(void) {
        MainWindow = new TWindow(NULL, "Hello, ObjectWindows!");
    }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    THelloApp Hello("HelloApp", hInst, hPrevInst, lpCmd, nShow);
    Hello.Run();
    return Hello.Status;
}
```

リスト20 test.cpp

```
#include <owl.h>

extern HANDLE _hInstance, _hPrev;
extern unsigned _psp;
extern PSTR _pszCmdline;
extern int _cmdShow;

class TApp: public TApplication {
public:
    TApp(void) : TApplication("TestApp", _hInstance, _hPrev,
        (LPSTR)((void _seg *)_psp + _pszCmdline), _cmdShow) {}
};

class TTestApp: public TApp {
public:
    void InitMainWindow(void) {
        MainWindow = new TWindow(NULL, "Hello, ObjectWindows!");
    }
};

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int)
{
    TTestApp App;
    App.Run();
    return App.Status;
}
```

要素となるものは，すべてTWindowsObjectを基本クラスとして持っているといってよいでしょう．反対に，TApplicationは目に見えないアプリケーション全体を管理するクラスです．TApplicationにウィンドウやダイアログボックスなどの視覚的要素を割り当てて，はじめて目に見えるアプリケーションが作成されるわけです．

さて，ウィンドウのタイトルを変えてみましょう．リスト19のようにプログラムできます．

ここでは，TApplicationからTHelloAppクラスを派生させて，InitMainWindowの動作を変更しています．変更しているといっても，ただウィンドウを生成するときに表示するタイトルを変更しているだけで，使っているウィンドウは相変わらず何もしないTWindowそのものです．TApplicationクラスのコンストラクタは引数を持っているので，それも引き継いで定義しなければなりません．

さて，ここでちょっと本題から外れます．TApplicationのコンストラクタが5つの引数を持っているので，それを受け継ぐクラスは必ずこの引数を持つコンストラクタを用意しなければなりません．そうしないと，WinMainに渡された引数を使ってTApplicationを初期化することができなくなってしまうからです．でも，クラスを継承するたびにコンストラクタを書かなければならないのもうっとおしいものです．なんとか手抜きをしようと考え出したのがリスト20です．

Borland C++ 3.0は，WinMainに渡している値を非公開変数として用意しています．_hInstance，_hPrev，_cmdShowはそれぞれWinMainの第1，2，4引数に相当します．_pszCmdlineだけは第3引数と同じでなくセグメント情報を追加してやらなければなりませんが，これらをTApplicationの初期化に使ってしまえば，WinMainの引数をコンストラクタに渡す必要はなくなります．

リスト20のWinMainでは，仮引数の名前が省略されていることに注目してください．C++では，使わない引数はその名前を記述しなくてもよいので，このように簡略化できるわけです．アプリケーションの名前も“TestApp”などに限定してしまえば，コンストラクタは引数を持たずにすみますから，派生クラスではもうコンストラクタを定義しなくてすみます．こうした定義をowl.hの末尾なり自分のヘッダにいれておけば，テストプログラムを記述するときにいちいち引数リストを並べたりコンストラクタを書かずにすみます．筆者はよくこの定義を使いますし，この解説の元のサンプルプログラムを作成するときにも利用しました．TApplicationを継承するたびにいちいちコンストラクタを継承するのも面倒です．一度使ってみてください．

では，本題に戻ります．今度は，ウィンドウそのものの動作を変えることにしましょう．そう，TWindowから新しいクラスを派生するのです．

リスト21は，マウスでクリックした位置に円を描画するツールです．TPointWindowはTWindowから派生されています．コンストラクタでは，デフォルトで使う半径（radius）と背景の色（color）を設定しています．このデフォルト値は，このクラスの中では変更することはありませんから描画するときに数値で指定してもかまいませんが，せっかくObjectWindowsで拡張によるプログラミングを行っているのですから，クラスを設計するときには常に拡張性に配慮しておくべきでしょう．

TPointWindowでは，PaintというTWindowの仮想関数をオーバー

Windows プログラミング用語集
ポインタ（Pointer） マウスの座標を示す．マウスカーソルとも呼ばれ，状態によって形状が変化する．

ライドしています．また，WM_LBUTTONDOWNとWM_RBUTTONDOWNというメッセージを受ける仮想関数をディスパッチ関数として定義しています．ObjectWindowsでは，Windowsメッセージを受けるメッセージ応答関数の名前にはメッセージ定数の識別子から下線を消去し，単語の先頭だけを大文字に変更したものを推奨しています．実際には，ディスパッチのインデックス値が正しければ関数名が違っていてもかまいません（わざわざ，そうする必要はありませんが）．

メッセージ定数にWM_FIRSTが加算されているのはWindowsメッセージ以外のメッセージもこの機構を使って同じように分岐できるようにするためです．WM_FIRSTは0，ID_FIRST(ダイアログコントロールなどで使われるID)は0x8000，NF_FIRST（チェックボックスやスクロールバーなどの通知メッセージ）は0x9000，CM_FIRST（メニューなどのコマンド)は0xA000が定義されていて，それぞれに同じ数値が使われていてもインデックス値としては異なる値として評価できるようにしているのです．

Paintの代わりに，WM_PAINTに対応したWMPaintを定義することもできますが，ObjectWindowsはウィンドウのスクロールに関する特別な処理をすることがありますからPaintをオーバーライドするようにしてください．

リスト 21　point.cpp

```
#include <owl.h>

class TPointWindow: public TWindow {
protected:
    int radius;
    DWORD color;
public:
    TPointWindow(PTWindowsObject AParent, LPSTR ATitle)
        : TWindow(AParent, ATitle) { radius = 5; color = RGB(255, 255, 0); }
    void Paint(HDC PaintDC, PAINTSTRUCT& ps)
    {
        LOGBRUSH LogBrush;
        LogBrush.lbStyle = BS_SOLID;
        LogBrush.lbColor = color;

        HBRUSH hBrush = CreateBrushIndirect(&LogBrush);
        SelectObject(PaintDC, hBrush);

        RECT rect;
        GetClientRect(&rect);
        Rectangle(PaintDC, rect.left, rect.top, rect.right, rect.bottom);
        DeleteObject(hBrush);
    }
    virtual void WMLButtonDown(RTMessage Msg) = [WM_FIRST + WM_LBUTTONDOWN]
    {
        HDC hdc = GetDC(HWindow);   // デバイスコンテキストの取得
        int x = Msg.LP.Lo;
        int y = Msg.LP.Hi;
        // マウスの位置に円を描画
        Ellipse(hdc, x - radius, y - radius, x + radius, y + radius);
        ReleaseDC(HWindow, hdc);    // デバイスコンテキストの解放
    }
    virtual void WMRButtonDown(RTMessage) = [WM_FIRST + WM_RBUTTONDOWN]
    {
        // ウィンドウを無効化して内容をクリアする
        InvalidateRect(HWindow, NULL, TRUE);
        UpdateWindow(HWindow);
    }
};

class TTestApp: public TApplication {
public:
    TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    // InitMainWindow のオーバーライド
    void InitMainWindow(void) {
        MainWindow = new TPointWindow(NULL, "Hello, Pointing Window!");
    }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    TTestApp App("TestApp", hInst, hPrevInst, lpCmd, nShow);
    App.Run();
    return App.Status;
}
```

WM_PAINTはWindowsがウィンドウの内容の再描画を必要と判断したときに送られてくるメッセージで，TWindow自身はWMPaintという関数でこれを処理しています．Paintは実際の塗りつぶしを行うようにWMPaintから呼び出されるものです．TWindowのPaint関数は何も行いません．この場合，WM_PAINTメッセージに対しては画面を真っ白に塗りつぶすだけになります．

Windowsでは，何かを描画するために必ずデバイスコンテキストが必要となります．Paint関数には，デバイスコンテキストとPAINTSTRUCT構造体型の情報が渡されますから，これを使って必要な描画を行うことができます．あるウィンドウに対するデバイスコンテキストは，そのウィンドウのクライアント領域と呼ばれる範囲にのみ描画するために使われます．クライアント領域とは，ウィンドウからタイトルバーやメニューなどを除いた描画領域のことで，範囲外の値を指定したところでクライアント領域の外に何かを描画することはできません．

ここでは，CreateBrushIndirectを使って黄色の塗りつぶしパターンをブラシとして作成しています．ブラシとは，Recangle(矩形)やEllipse(楕円)などの塗りつぶしで使われるパターンのことです．SelectObjectで実際に作成したブラシを選択して，GetClientRectで得られたクライアント領域をRectangleで塗りつぶしています．これによって，Paintは描画要求に対し

Windows プログラミング用語集
ホットスポット（Hot spot）　マウスがディスプレ上で示す座標のこと．通常はマウスカーソルの矢印の先端や十字の中心が示す座標がホットポイント．

て黄色で塗りつぶすという処理を実現しています（第2引数として渡されたPAINTSTRUCT構造体の中には最低限書き変えればよい矩形領域が入っているのですが，このプログラムではあまりうまくない方法で背景色を塗りつぶしているのでこのような手法を取っています）．必要のなくなったブラシはDeteleObjectで解放します．

WMLButtonDownは，左ボタンが押されたときに送られるWM_LBUTTONDOWNメッセージによって呼び出されます．ここでは，押されたボタンの位置にEllipseで円を描画しています．ここではブラシを定義していないので，白抜きの円になります．Paint関数とは違ってデバイスコンテキストが与えられないため，自分でデバイスコンテキストを確保しなければなりません．このためにGetDCというWindows APIを使っています．GetDCで得たデバイスコンテキストは，使い終わったらすぐにReleaseDCで解放してやらないといけません．Windowsにはデバイスコンテキストが5つしかないので，あるウィンドウがそれを確保したままになっていると，ほかのプログラムがデバイスコンテキストを取得できなくなってしまうからです．ボタンが押された情報をどこにも格納していないので，Paintなどでクライアント領域が更新されると描画した円は消されてしまいます．

WMRButtonDownは，右ボタンが押されたときに送られるWM_RBUTTONDOWNメッセージによって呼び出されます．ここでは，InvalidateRectでそのウィンドウのクライアント領域全体を無効化しています．Windowsは，あるウィンドウのクライアント領域が無効化されたことを知らされると，そのウィンドウに対してクライアント領域を書き直すようWM_PAINTメッセージを送ります．これで，右ボタンが押されたときにクライアント領域全体を黄色で塗りつぶすという処理を実現しているわけです．

ウィンドウを拡張する

繰り返し述べているとおり，ObjectWindowsのプログラミングは拡張する形で行われます．さきほどのアプリケーションにメニューコマンドを追加してみましょう．

リスト22では，リスト21のTPointWindowから円の大きさと背景色を変更できるようなクラスTPointWinを派生しています．TTestAppのInitMainWindowは，TPointWindowでなくTPointWinのインスタンスを割り当てています．

リソースとしてのメニューは，ウィンドウに割り当てられるものです．ObjectWindowsではメニューリソースを割り当てるためにTWindowのメンバ関数AssignMenuを使います．ここでは，派生したコンストラク

リスト 22　pointwi_.cpp

```
#include <owl.h>
#include "pointwin.h"

    ......

class TPointWin: public TPointWindow {
public:
    TPointWin(PTWindowsObject AParent, LPSTR ATitle)
        : TPointWindow(AParent, ATitle) { AssignMenu("POINTWIN"); }
    // 画面の再描画
    void Repaint(void) {
        InvalidateRect(HWindow, NULL, TRUE);
        UpdateWindow(HWindow);
    }
    // Radius (半径) メニュー
    virtual void Radius2(RTMessage) = [CM_FIRST + CM_RADIUS2]
        { radius = 2; }
    virtual void Radius5(RTMessage) = [CM_FIRST + CM_RADIUS5]
        { radius = 5; }
    virtual void Radius10(RTMessage) = [CM_FIRST + CM_RADIUS10]
        { radius = 10; }
    // Color メニュー
    virtual void ColorYellow(RTMessage) = [CM_FIRST + CM_COLORYELLOW]
        { color = RGB(255, 255, 0); Repaint(); }
    virtual void ColorBlue(RTMessage) = [CM_FIRST + CM_COLORBLUE]
        { color = RGB(0, 0, 255); Repaint(); }
    virtual void ColorRed(RTMessage) = [CM_FIRST + CM_COLORRED]
        { color = RGB(255, 0, 0); Repaint(); }
    // About メニュー
    virtual void About(RTMessage) = [CM_FIRST + CM_ABOUT]
        { MessageBox(HWindow, "PointWindow!", "About", MB_OK); }
};

class TTestApp: public TApplication {
public:
    TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    // InitMainWindow のオーバーライド
    void InitMainWindow(void) {
        MainWindow = new TPointWin(NULL, "Hello, Pointing Window!");
    }
};
    ......
```

リスト 23　pointwin.rc

```
#include <windows.h>
#include "pointwin.h"

POINTWIN MENU
BEGIN
    POPUP "&Radius"
        BEGIN
            MENUITEM "&2 dots", CM_RADIUS2
            MENUITEM "&5 dots", CM_RADIUS5
            MENUITEM "&10 dots", CM_RADIUS10
        END
    POPUP "&Color"
        BEGIN
            MENUITEM "&Yellow", CM_COLORYELLOW
            MENUITEM "&Blue", CM_COLORBLUE
            MENUITEM "&Red", CM_COLORRED
        END
    MENUITEM "&About", CM_ABOUT
END

POINTWIN DIALOG 18, 18, 142, 92
CAPTION "About"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
BEGIN
    LTEXT "This is Pointing Window!", -1, 28, 14, 85, 8
    LTEXT "Created by", -1, 51, 29, 39, 8
    LTEXT "Motohisa Ohno.", -1, 43, 44, 56, 8
    PUSHBUTTON "OK", IDOK, 59, 69, 24, 14, WS_CHILD | WS_VISIBLE | WS_TABSTOP
END
```


Windows プログラミング用語集

ホットリンク（Hot link）　個々のアプリケーションの2つのファイル間の動的な関係．DDEによって作成される．

タでリスト23で示されたメニューリソースを割り当てています．背景色が変更されたときのためにクライアント領域の再描画を促すメンバ関数Repaintを作成しています．基本クラスのWMRButtonDownを呼び出してもよいのですが，ここでは「WMRButtonDownを処理」してほしいわけではなく，「画面を再描画」してほしいわけですから，そのための関数を用意しました．もっとも，基本クラスで拡張性を考慮しておけばよかったわけです．

メニューコマンドは，WM_COMMANDメッセージを生成しますからWMCommandというディスパッチ関数を定義することができます．しかし，前述したようにObjectWindowsでは個々のメニューコマンドに対してもダイナミックディスパッチ機構を使うことができますから，ここではそれを利用しています．

通常，TWindowsObjectから派生したクラスがWM_COMMANDを処理しないとTWindowsObjectクラスのメンバ関数WMCommandが受け取ったコマンドの種類によってディスパッチ関数を呼び出します．ここでは，リスト24で示されているメニューコマンドに対してColorYellow, ColorBlue, ColorRed, Radius2, Radius5, Radius10, Aboutというディスパッチ関数を定義しています．インデックス値にはメニューコマンドに対するCM_FIRSTを加算しておくことを忘れないでください．

基本クラスのWMCommandがディスパッチ処理を行うことがわかれば，メニューコマンドによってWMCommandで処理するかディスパッチ関数で処理するかを使い分けることもできるでしょう．これは，リスト25のようなプログラムで実現できます．

ウィンドウのスタイルや属性を変更するにはどうすればよいでしょうか．ウィンドウを作成するときの位置やシステムメニューやフレームのスタイル（CreateWindowで設定する内容）を変更するだけならば，TWindow::SetupWindowをオーバーライドしてウィンドウ属性をあらわすデータメンバAttrを変更します．たとえば，ポップアップウィンドウにしたい場合は，リスト26のようにします．ウィンドウを縮小化したときのアイコンやデフォルトのカーソル形状など，ウィンドウクラスを変更しなければならないときはTWindow::GetWindowClassをオーバーライドします（リスト27）．

デフォルトでは，ウィンドウクラス名にはアプリケーションを生成したときのアプリケーション名が使われます．複数のウィンドウクラスを使う場合は，ウィンドウごとにGetClassNameをオーバーライドしなければなりません．

また，いずれの場合も必ず基本クラスのメンバ関数（SetupWindow, GetWindowClass）を呼び出さなけ

リスト 24　pointwin.h

```
#define CM_RADIUS2      200
#define CM_RADIUS5      201
#define CM_RADIUS10     202
#define CM_COLORYELLOW  210
#define CM_COLORBLUE    211
#define CM_COLORRED     212
#define CM_ABOUT        220
```

リスト 25

```
class TDerivedWindow: public TWindow {
public:
    ......
    virtual void WMCommand(RTMessage Msg) = [WM_FIRST + WM_COMMAND]
    {
        if  (Msg.WParam はまとめて処理したいメニューコマンド)  {
            コマンドの処理
        } else {
            // 基本クラスの WMCommand を呼び出すと
            // ダイナミックディスパッチが処理される
            TWindow::WMCommand(Msg);
        }
    }
```

リスト 26

```
class TDerivedWindow: public TWindow {
public:
    ......
    void SetupWindow(void)
    {
        TWindow::SetupWindow();
        Attr.Style |= WS_POPUPWINDOW | WS_CAPTION;
        Attr.X = 100;  // 左上のX座標
        Attr.Y = 50;   // 左上のY座標
        Attr.W = 200;  // ウィンドウの水平方向の大きさ
        Attr.Y = 200;  // ウィンドウの垂直方向の大きさ
    }
```

リスト 27

```
class TDerivedWindow: public TWindow {
public:
    ......
    void GetWindowClass(WNDCLASS& AWndClass)
    {
      TWindow::GetWindowClass(AWndClass);
      AWndClass.hIcon = LoadIcon(GetModule()->hInstance, "MYICON");
    }
```

Windowsプログラミング用語集

マッピングモード（Mapping mode）　プログラムから送られてきた論理座標をマッピングモードにあわせて変換し，スクリーンに表示する．

リスト 28 dialapp.cpp

```
#include <owl.h>

class TTestApp: public TApplication {
public:
    TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    void InitMainWindow(void) {
        // ダイアログボックスをメインウィンドウに指定
        MainWindow = new TDialog(NULL, "DIALAPP");
    }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    TTestApp App("TestApp", hInst, hPrevInst, lpCmd, nShow);
    App.Run();
    return App.Status;
}
```

リスト 29 dialapp.rc

```
#include <windows.h>

DIALAPP DIALOG 18, 18, 142, 92
CAPTION "Helo, DialogApp!"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
BEGIN
    PUSHBUTTON "OK", IDOK, 49, 68, 43, 14, WS_CHILD | WS_VISIBLE | WS_TABSTOP
    LTEXT "This is a Dialog Application.", -1, 22, 13, 97, 8
    LTEXT "Created by", -1, 51, 27, 40, 8
    LTEXT "Motohisa Ohno.", -1, 43, 42, 56, 8
END
```

ればなりません．これらのメンバ関数ではウィンドウを構築するために必要な処理を行っています．呼び出しを忘れるとデータメンバが初期化されないままになってしまいます．

さて，About関数ではGetApplication()という関数が使われています．これは，TWindowsObjectのメンバ関数で，現在実行中のアプリケーションオブジェクト（TApplication型のインスタンス）へのポインタを返します．ExecDialogは，TApplicationの基本クラスTModuleのメンバ関数でモードつきダイアログボックスを実行します．ダイアログボックスのクラスTDialogは，親ウィンドウへのポインタとダイアログリソース名を引数とするコンストラクタを持ち，生成されたインスタンスはExecDialogによって実際に呼び出されて画面に表示されます．TDialogクラスはIDOKを受け取るとダイアログボックスを終了しますが，それ以外は特に何も行わないクラスです．TDialogの拡張については後述します．

ExecDialogと同様のメンバ関数にMakeWindowがあります．MakeWindowは，ウィンドウをアプリケーション上に生成させるためのものです．TApplicationは，その初期化の段階で仮想関数InitMainWindowでMainWindowに割り当てられたウィンドウをMakeWindowに渡すことで，アプリケーションのメインウィンドウをオープンしています．複数のダイアログボックスやウィンドウを使うようになると，ExecDialogやMakeWindowがよく使われるようになります．

ダイアログボックスをメインウィンドウにする

メインウィンドウには，ダイアログボックスを指定することもできます．ダイアログボックスが，ウィンドウの一種であることを考えればこれはごく自然なことです．また，ダイアログボックスもウィンドウもTWindowsObjectという基本クラスから派生しているためどちらのインスタンスもMainWindowに指定できるのです．

リスト28がダイアログボックスをメインウィンドウに割り当てたプログラム例で，リスト29がダイアログボックスのためのリソースです．

TDialogでは，第2引数がキャプションでなくダイアログリソースを指します．ここでは，文字列“DIALAPP”でリソースを指定しています．デフォルトでは，TDialogはIDOKというメッセージが渡されるとダイアログボックスをクローズします．このプログラムは，このコマンドだけを利用したごく簡単なプログラムです．

TDialogもTWindowを拡張したのと同じように拡張することができます．とくに，ダイアログボックスは張り付けられたコントロールからのメッセージを受け付けられます．TDialogから派生したクラスからダイアログボックスに張り付けられているコントロールを操作したい場合は，派生クラスの中に適当なデータメンバを用意しておきます．たとえば，リストボックスコントロールを使う場合は，そのリス

リスト 30

```
#include <listbox.h>
#include "myheader.h"

class TMyDialog: public TDialog {
    PTListBox list;
public:
    TMyDialog(PTWindowsObject AParent, LPSTR AName)
        : TDialog(AParent, AName) {
        list = new TListBox(this, ID_MYLISTBOX);
    }
    void additem(const char *item) {
        list->AddString(item);
    }
    ......
};
```


Windows プログラミング用語集
メタファイル（Metafiles）　GDI関数をバイナリフォーマットにコード化して集めたものを指す．

トボックスに文字列を設定したり選択したデータを取得したいはずです．この場合は，TListBoxへのポインタ（PTListBox）を派生したクラスにデータメンバとして用意するのです．コンストラクタでは，このデータメンバにリソースで定義されたコントロールのIDを使ってそのコントロールのインスタンスを割り当てます．たとえば，リスト30のようになります．

Windows APIを使う場合は，いちいちSendMessageでパラメータを渡してやらないといけませんが，ObjectWindowsでは派生したクラスの中にコントロールに対応したデータメンバを用意することで，直接コントロールを操作できるような間隔でプログラミングできるようになります．

TDialogから派生したクラスをメインウィンドウにするとコントロールを必要とする処理が簡単に実現できます．リスト31，32，33はダイアログボックスをメインウィンドウにして整数電卓を作成したものです．

TDialogから派生したTCalcDialogクラスのコンストラクタでは，スタティックテキストに対するTStaticクラスのインスタンスを生成しています．電卓の入力値や計算値を表示するためにダイアログボックスのスタティックコントロールを使いたいので，このインスタンスが必要となるのです．

このダイアログボックスでは，スタティックコントロール以外にはボタンしかありません．ボタンはダイアログボックスに対してコマンドメッセージを送りますが，ここでは前述した手法によってコマンドをWMCommandで処理するかディスパッチ関数で処理するかを使い分けています．［C］ボタンと［=］ボタンに対する処理をディスパッチ関数（ClearとEqual）で処

リスト 31 calc.cpp

```
#include <owl.h>
#include <static.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "calc.h"

#define MAXLEN 10
typedef unsigned long ULONG;

class TCalcDialog: public TDialog {
    char number[MAXLEN + 1];
    BOOL first;
    int op_id;
    ULONG stack;
    PTStatic display;
    int radix;

public:
    TCalcDialog();
    void clearall(void);
    virtual void Clear(RTMessage Msg) = [ID_FIRST + ID_CLEAR];
    virtual void Equal(RTMessage Msg) = [ID_FIRST + ID_EQUAL];
    virtual void WMCommand(RTMessage Msg) = [WM_FIRST + WM_COMMAND];
};

TCalcDialog::TCalcDialog() : TDialog(NULL, "CALC")
{
    display = new TStatic(this, ID_DISPLAY, MAXLEN + 1);
    radix = 10;
    clearall();
}

void TCalcDialog::clearall(void)
{
    strcpy(number, "0");
    stack = 0;
    first = TRUE;
    op_id = ID_PLUS;
    display->SetText(number);
}

void TCalcDialog::Clear(RTMessage)
{
    clearall();
}

void TCalcDialog::Equal(RTMessage)
{
    if (first)
        return;
    ULONG temp = strtoul(number, NULL, radix);
    switch (op_id) {
    case ID_DIVIDE:
        if (temp == 0L) {
            MessageBox(HWindow, "Divide by Zero", "Divide error!", MB_OK);
            clearall();
        }
        stack /= temp;
        break;
    case ID_MULTIPLY:   stack *= temp; break;
    case ID_PLUS:       stack += temp; break;
    case ID_MINUS:      stack -= temp; break;
    }
    sprintf(number, "%lu", stack);
    first = TRUE;
    display->SetText(number);
}

void TCalcDialog::WMCommand(RTMessage Msg)
{
    if (ID_BUTTON0 <= Msg.WParam && Msg.WParam <= ID_BUTTON9) {
        // 数値の入力
        if (first || strcmp(number, "0") == 0) {
            number[0] = '0' + (Msg.WParam - ID_BUTTON0);
            number[1] = 0;
        } else if (strlen(number) < MAXLEN) {
            char buf[2];
            buf[0] = '0' + (Msg.WParam - ID_BUTTON0);
            buf[1] = 0;
            strcat(number, buf);
        }
        first = FALSE;
        display->SetText(number);
    } else if (ID_DIVIDE <= Msg.WParam && Msg.WParam <= ID_MINUS) {
        if (!first)
            stack = strtoul(number, NULL, radix);
        first = TRUE;
        op_id = Msg.WParam;
    } else {
        TDialog::WMCommand(Msg);
    }
}

class TCalcApp: public TApplication {
public:
    TCalcApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    void InitMainWindow(void) {
        // ダイアログボックスをメインウィンドウに指定
        MainWindow = new TCalcDialog;
    }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    TCalcApp App("CalcApp", hInst, hPrevInst, lpCmd, nShow);
    App.Run();
    return App.Status;
}
```

理し，数字ボタンと演算子ボタンはWMCommandで処理しています．ボタンコントロールのIDをディスパッチ関数に指定するためには，ボタンコントロールのIDにID_FIRSTを加算します．

このプログラムの処理について少しだけ解説します．

電卓の初期化を行う［C］ボタンに対するClear関数は表示文字列を“0”にして，内部のスタックをクリアします．また，演算子の状態を［+］の状態にしておきます．Equal関数は，表示文字列を数値に変換し，内部スタックの値を保持している演算子によって計算します．除算の場合，除数が0のときはあらかじめチェックして計算を行わないようにします．結果を表示し，その値は内部スタックに保持しておき連続計算に備えます．

WMCommandでは，数字ボタンが押されたらそれを表示文字列に追加します．最初の文字の入力ではその文字を表示文字列とします．演算子ボタンが押されたら，表示文字列を数値に変換し内部スタックに代入し，演算子IDは［=］ボタンが押されたときのために保持しておきます．それ以外のコマンドについては，基本クラスのWMCommandにまかせてディスパッチ関数を呼び出すようにしています．

計算は，unsigned long（符号なし整数）で行われるので負の値や小数点は表現できません．基数はデータメンバradixを使っているので，ボタンを増やして16進電卓にしたりすることができるでしょう．

定義済みのクラスを使う

ObjectWindowsには，あらかじめいろいろなクラスが定義されていますから，それを使ってみることにしましょう．リスト34は，リスト19においてメインウィンドウにTWindowでなくTEditWindowのインスタンスを使った例です．

ファイルの入出力ができないのでたいした役には立ちませんが，これだけでスクロールバーと編集領域をもつアプリケーションが作成できました．TEditWindowは，カット&ペーストなどの編集機能のほかに検索機能も備えているのですが，ここではそれらを起動するための方法が定義されていないために使えません．TEditWindowを使うためのリソースはObjectWindowsとともに提供されていますからそれを使えばよいのですが，せっかくですからより機能の高いTFileWindowクラスを使ってみましょう．

TFileWindowはTEditWindow

リスト32　calc.h

```
#define ID_DISPLAY  200
#define ID_DIVIDE   201
#define ID_MULTIPLY 202
#define ID_PLUS     203
#define ID_MINUS    204
#define ID_CLEAR    206
#define ID_EQUAL    205
#define ID_BUTTON1  301
#define ID_BUTTON2  302
#define ID_BUTTON3  303
#define ID_BUTTON0  300
#define ID_BUTTON4  304
#define ID_BUTTON5  305
#define ID_BUTTON6  306
#define ID_BUTTON7  307
#define ID_BUTTON8  308
#define ID_BUTTON9  309
```

リスト33　calc.rc

```
#include <windows.h>
#include "calc.h"

CALC DIALOG 18, 18, 110, 102
CAPTION "Calculator"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
BEGIN
    CONTROL "0", ID_BUTTON0, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 13, 78, 18, 14
    CONTROL "1", ID_BUTTON1, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 13, 60, 18, 14
    CONTROL "2", ID_BUTTON2, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 35, 60, 18, 14
    CONTROL "3", ID_BUTTON3, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 57, 60, 18, 14
    CONTROL "4", ID_BUTTON4, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 13, 42, 18, 14
    CONTROL "5", ID_BUTTON5, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 35, 42, 18, 14
    CONTROL "6", ID_BUTTON6, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 57, 42, 18, 14
    CONTROL "7", ID_BUTTON7, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 13, 24, 18, 14
    CONTROL "8", ID_BUTTON8, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 35, 24, 18, 14
    CONTROL "9", ID_BUTTON9, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 57, 24, 18, 14
    CONTROL "C", ID_CLEAR, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 35, 78, 18, 14
    CONTROL "=", ID_EQUAL, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 57, 78, 18, 14
    CONTROL "-", ID_MINUS, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 79, 78, 18, 14
    CONTROL "+", ID_PLUS, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 79, 60, 18, 14
    CONTROL "*", ID_MULTIPLY, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 79, 42, 18, 14
    CONTROL "/", ID_DIVIDE, "BUTTON", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 79, 24, 18, 14
    CONTROL "0", ID_DISPLAY, "STATIC", SS_RIGHT | WS_CHILD | WS_VISIBLE | WS_GROUP, 14, 8, 83, 8
    CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD | WS_VISIBLE | WS_GROUP, 10, 4, 92, 15
END

```

リスト34　edit.cpp

```
#include <owl.h>
#include <editwnd.h>

class TTestApp: public TApplication {
public:
    TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
        :TApplication(AName, Inst, Prev, Cmd, Show) {}
    void InitMainWindow(void) {
        MainWindow = new TEditWindow(NULL, "Hello, EditWindow!");
    }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
    TTestApp App("TestApp", hInst, hPrevInst, lpCmd, nShow);
    App.Run();
    return App.Status;
}
```


Windows プログラミング用語集
ラジオボタン（Radio botton）　ダイアログボックス内などで，複数の選択肢から排他的な選択をするボタンで，オプションボタンとも呼ばれる．

から派生したクラスで，検索機能のほかにファイルとの入出力機能も備えています．これで，エディタを作成するわけですが必要なものはすべて用意されていますから何も難しいことはありません．リスト35，36がそのすべてです．

TEditorWindowはTFileWindowから派生し，そのコンストラクタでウィンドウのメニューをFILEMENU.RCで定義されている“FileCommands”に割り当てています．TFileWindowが自分でFileCommandsを割り当てていないのは，ほかのメニューリソースが割り当てられる場合に対応するためです．

TFileWindowは，その内部で次のリソースを使用しています．

SD_INPUTDIALOG (inputdia.rc)

一般的な1行入力を行うダイアログボックスです．TInputDialogクラスがこのダイアログボックスの処理を行います．

SD_FILEOPEN (filedial.rc)

オープンするファイルを選択するためのダイアログボックスです．次のSD_FILESAVEとともにTFileDialogがこのダイアログボックスの処理を行います．

SD_FILESAVE (filedial.rc)

ファイルを保存するときのファイル名を選択するためのダイアログボックスです．

SD_SEARCH (stdwnds.rc)

検索する文字列を指定するためのダイアログボックスです．次のSD_REPLACEとともにTSearchDialogがこのダイアログボックスの処理を行います．

SD_REPLACE (stdwnds.rc)

置換する2つの文字列とオプションを指定するためのダイアログボックスです．

リスト35　editor.cpp

```
#include <owl.h>
#include <filewnd.h>

class TEditorWindow: public TFileWindow {
public:
        TEditorWindow(PTWindowsObject AParent, LPSTR ATitle, LPSTR AFileName)
                :TFileWindow(AParent, ATitle, AFileName) {
                AssignMenu("FileCommands");
        }
};

class TTestApp: public TApplication {
public:
        TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
                :TApplication(AName, Inst, Prev, Cmd, Show) {}
        void InitMainWindow(void) {
                MainWindow = new TEditWindow(NULL, "Hello, EditWindow!", "");
        }
        void InitInstance(void) {
                TApp::InitInstance();
                HAccTable = LoadAccelerators(hInstance, "FileCommands");
        }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
        TTestApp App("TestApp", hInst, hPrevInst, lpCmd, nShow);
        App.Run();
        return App.Status;
}
```

リスト36　editor.rc

```
#include <windows.h>
#include <owlrc.h>

/* 入力ダイアログ */
#include <inputdia.dlg>
/* ファイル選択ダイアログ */
#include <filedial.dlg>
/* 検索・置換ダイアログ */
#include <stdwnds.dlg>
/* ファイルアプリケーションのアクセラレータ */
#include <fileacc.rc>
/* ファイルアプリケーションのメニュー */
#include <filemenu.rc>
```

FILECOMMANDS (fileacc.rc)

デフォルトのアクセラレータ（キーボードのショートカット機能）の割り当てです．

FILECOMMANDS (filemenu.rc)

デフォルトのファイルウィンドウに対するメニューです．

TTestAppクラスでは新たに仮想関数のInitInstanceをオーバーライドして，アクセラレータを初期化しています．InitInstanceではアプリケーションに属する初期化を行います．前述したMakeWindowでアプリケーションのメインウィンドウをオープンするのも基本クラスのInitInstanceが行っています．したがって，オーバーライドした場合は必ず基本クラスの関数も呼び出しておかなければなりません．そうでないと必要な初期化が行われなくなってしまうからです．

TFileWindowクラスは，それだけで使うならば既存のメモ帳（Notepad）を使うほうがよいでしょう．しかし，このクラスを自分のアプリケーションに組み入れられるというのは大きなメリットです．次は，Windows 3.0のMDIとTFileWindowを使ってマルチファイルを扱えるアプリケーションを作成してみます．

Windowsプログラミング用語集
ラスタフォント（Raster fonts）　特定のサイズでデザインされ，ドットで構成されるフォントのこと．サイズを変更するとギザギザになる．

リスト 37　mdiedit.cpp

```
#include <owl.h>
#include <filewnd.h>
#include <mdi.h>

class TFrameWindow: public TMDIFrame {
public:
        TFrameWindow(LPSTR ATitle, LPSTR Menu)
                :TMDIFrame(ATitle, Menu) { ChildMenuPos = 3; }
        virtual void NewFile(RTMessage Msg) = [CM_FIRST + CM_MDIFILENEW];
        virtual void OpenFile(RTMessage Msg) = [CM_FIRST + CM_MDIFILEOPEN];
};

void TFrameWindow::NewFile(RTMessage Msg)
{
        GetApplication()->MakeWindow(new TFileWindow(this, "File", ""));
}

void TFrameWindow::OpenFile(RTMessage Msg)
{
        char FileName[80] = "*.*";
        if (GetApplication()->ExecDialog(
                        new TFileDialog(this, SD_FILEOPEN, FileName)) == IDOK)
            GetApplication()->MakeWindow(new TFileWindow(this, "File", FileName));
}

class TTestApp: public TApplication {
public:
        TTestApp(LPSTR AName, HANDLE Inst, HANDLE Prev, LPSTR Cmd, int Show)
                :TApplication(AName, Inst, Prev, Cmd, Show) {}
        void InitMainWindow(void) {
                MainWindow = new TFrameWindow("MDI Editor", "MDIEDIT");
        }
        void InitInstance(void) {
                TApp::InitInstance();
                HAccTable = LoadAccelerators(hInstance, "FileCommands");
        }
};

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmd, int nShow)
{
        TTestApp App("TestApp", hInst, hPrevInst, lpCmd, nShow);
        App.Run();
        return App.Status;
}
```

リスト 38　mdiedit.rc

```
#include <windows.h>
#include <owlrc.h>

/* 入力ダイアログ */
#include <inputdia.dlg>
/* ファイル選択ダイアログ */
#include <filedial.dlg>
/* 検索・置換ダイアログ */
#include <stdwnds.dlg>
/* ファイルアプリケーションのアクセラレータ */
#include <fileacc.rc>

MDIEDIT MENU
BEGIN
    POPUP "&File"
        BEGIN
            MENUITEM "&New", CM_MDIFILENEW
            MENUITEM "&Open", CM_MDIFILEOPEN
            MENUITEM "&Save", CM_FILESAVE
            MENUITEM "Save &as...", CM_FILESAVEAS
            MENUITEM SEPARATOR
            MENUITEM "E&xit", CM_EXIT
        END
    POPUP "&Edit"
        BEGIN
            MENUITEM "&Undo¥aGRPH+BS", CM_EDITUNDO
            MENUITEM SEPARATOR
            MENUITEM "&Cut¥aSHIFT+DEL", CM_EDITCUT
            MENUITEM "C&opy¥aCTRL+INS", CM_EDITCOPY
            MENUITEM "&Paste¥aSHIFT+INS", CM_EDITPASTE
            MENUITEM "&Delete¥aDEL", CM_EDITDELETE
            MENUITEM "C&lear all¥aCTRL+DEL", CM_EDITCLEAR
        END
    POPUP "&Search"
        BEGIN
            MENUITEM "&Find...", CM_EDITFIND
            MENUITEM "&Replace...", CM_EDITREPLACE
            MENUITEM "&Next¥aF3", CM_EDITFINDNEXT
        END
    POPUP "&Window"
        BEGIN
            MENUITEM "&Cascade", CM_CASCADECHILDREN
            MENUITEM "&Tile", CM_TILECHILDREN
            MENUITEM "&Arrange Icons", CM_ARRANGEICONS
            MENUITEM "C&lose all", CM_CLOSECHILDREN
        END
END
```

MDIを使う

Windows 3.0には，マルチウィンドウを扱うためのMDIという仕組が用意されています．ObjectWindowsでもMDIのためのクラスが用意されていますから，これを使ってみます．

MDIを使うためには，2つのウィンドウを使います．1つはクライアントウィンドウを呼ばれるもので，フレームウィンドウのクライアント領域と同じ大きさのウィンドウです．クライアントウィンドウは，Windows 3.0が持っているウィンドウクラス“MDI-CLIENT”を使うことで実際のMDIの処理のほとんどを引き受けますが，プログラムから直接アクセスされることはあまりありません．もう1つはフレームウィンドウで，クライアントウィンドウとアプリケーションへの要求の受け渡し役をはたすメインウィンドウとなります．

実際にMDIを使ったプログラム例をリスト37，38に示します．

TMDIFrameは，キャプションとメニューを引数としてとります．フレームウィンドウが子ウィンドウになることはないので，親ウィンドウへのポインタはありません．TMDIFrameから派生したクラスTFrameWindowでは，NewFileとOpenFileという仮想関数をDD関数として定義しています．これ以外のコマンドは自動的にTMDIFrameが処理するか，あるいはオープンしたファイルウィンドウに渡されます．

TFrameWindowのメンバ関数NewFile，OpenFileでは，前述したMakeWindowで新たなウィンドウを追加しています．追加しているウィ

Windowsプログラミング用語集
リソース（Resource）　各Windowsプログラムで使われるデータやオブジェクトのこと．プログラムから独立することで共有化をはかる．

リスト 39　heaptest.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
    char *p;
    int i;
    for (i = 0; i < 5000; i++) {
        p = malloc(16);
        malloc(1);        // (*)
        free(p);
    }
    printf("%ld¥n", (long)(clock() / CLOCKS_PER_SEC));
}
```

図3

malloc(1);	BC++ 3.0	MS-C 6.0
なし	1秒	1秒
あり	1秒	29秒

※PC-9801RA2

ンドウは，TFileWindowそのものです．OpenFileではTFileDialogというファイル選択用のダイアログボックスを呼び出して，既存ファイルを選択させています．フレームウィンドウが行うことはこれだけです．

後は，アプリケーションのメインウィンドウにTFrameWindowのインスタンスを使う以外，EDITORアプリケーションとほとんど同じです．ただし，リソースファイルはC++のように継承が使えないので新たなメニュー（MDIEDIT）を作成しています．

いかがですか．リソースを除けばほんの40行あまりのプログラムでマルチファイルエディタが完成してしまいました．マルチファイルエデェタそのものは，さして目新しいものでもありませんし機能は多くありませんが，自分自身のアプリケーションにテキストファイルを編集するような部分があれば，このTFileWindowやそこから派生したクラスをごく自然に自分のアプリケーションに組み入れることができるわけです．

コンパイラの評価について

ObjectWindowsのまとめとしては不適当かもしれませんが，最近コンパイラの評価について気になっていることを書いておきます．

Borland C++ 3.0の最大の目玉は最適化機能のようにいわれているようですが，よくいわれるようにどんなにコンパイラががんばってもアルゴリズムそのものが変わらない限りプログラムの速度がそんなに向上することはありません．

リスト39を見てください．なんの変哲もないヒープメモリを使う簡単なプログラムですが，Borland C++ 3.0とMicrosoft C 6.0(7.0では未調査)で実行すると図3のような結果になります．極端な例かもしれませんが，たった1行の存在がこれだけ実行速度に影響することもあるわけです．これは，別にMS-C 6.0のヒープ関数に問題があるわけではなく単に状況によって不得手な面が現れてしまったということでしょう．Borland C++でも別の状況では同様のことが生じるかもしれません．

ライブラリ関数や内部仕様に依存する問題には，最適化オプションはほとんど効果がありません．ベンチマークテストというものがよく行われますが，無駄だらけのプログラムをいくら最適化できたところで実際のアプリケーションに役立つとは限りません．ある記事ではライブラリ関数の評価と称して，個々の関数を繰り返し呼び出して実行時間を計測していましたが，関数が組み合わされて使われるものだということをまったく配慮していないもので開いた口が塞がりませんでした．

ベンチマークテストは客観的なデータを得るためには必要ですが，それがコンパイラのすべてではありません．しかも，少し取捨選択をするだけで主観的なデータとなってしまうこともあります．また，コンパイラの内部仕様が実行速度に影響することもあります．そうした点に配慮しなければ本当に役立つ資料とはなり得ないでしょう．

もし実行速度を向上させたいのであれば，Borland C++ 3.0をはじめ他の処理系でもサポートされているプロファイラを使って，本当に問題になっている箇所を突き止めるべきでしょう．

開発環境は，コンパイル速度，生成コードの品質，開発ツール，提供されるライブラリ，デバッガなどさまざまな要素によって決められます．また，使う人によってどのようなものが重視されるかも異なります．すべてのこうした記事を信用するなとはいいませんが，そのテストが本当にコンパイラの仕様や実際に使われる環境にまで配慮されたものであるかどうか見極める目を持ちたいものです．

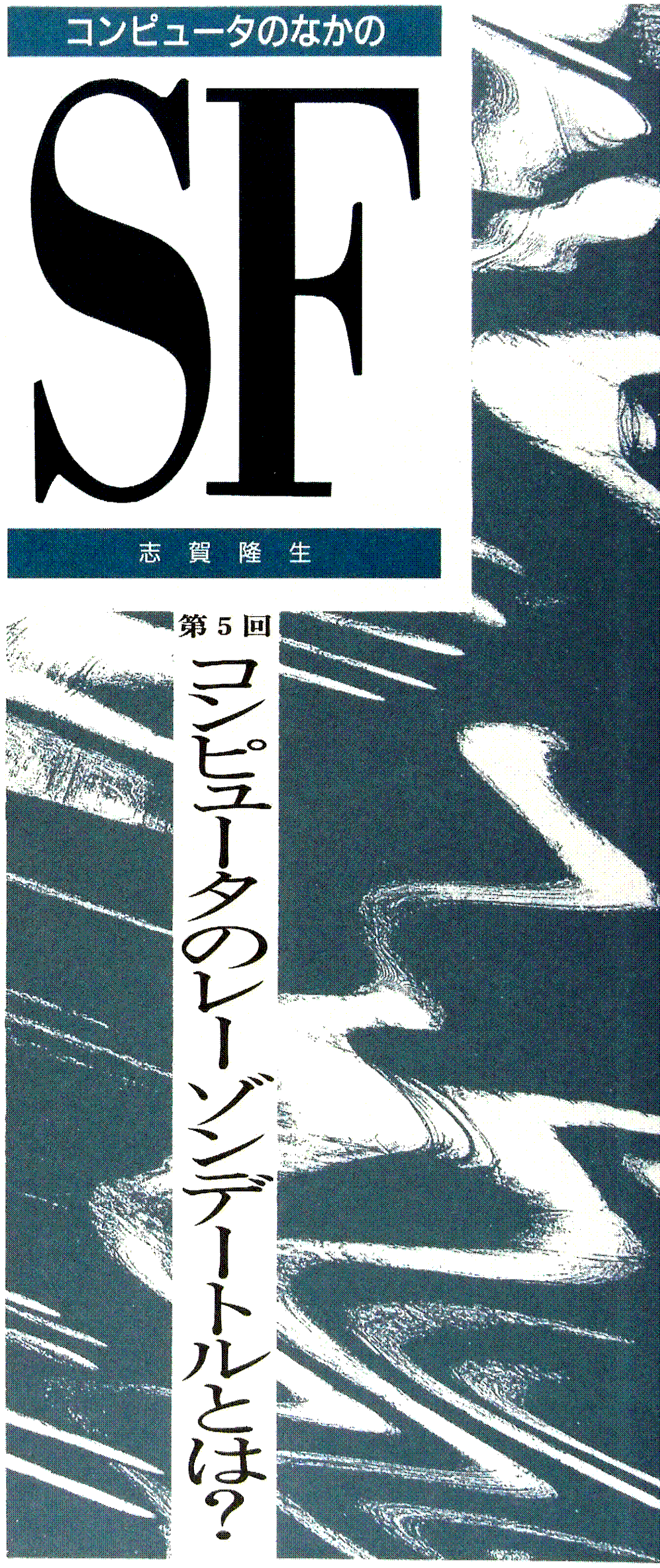
コンピュータのなかの

SF

志 賀 隆 生

第5回

コンピュータのレーゾンデートルとは?



前回は「ドラッグを楽しむコンピュータ」と題して，いまは懐かしのサンリオSF文庫中の異色作『H・A・R・L・I・E』のさわりの部分を紹介した．

主人公は，ニューラルネットワークとファジー理論をベースにした近未来のAIコンピュータ「ハーリイ」．外界の知識を吸収するにしたがって，ハーリイは高度な判断能力とともに自意識を持つようになる．ちょうど人間の心の成長に似て，ハーリイはコンピュータが乳幼児期から少年期へ，少年期から青年期へと成長していくのだ．もちろん，その成長のスピードは人間と比較にならないほど早い．そして，人間でいえば少年期から青年期へ進むころ，同じように反抗期を迎える．知識の拡大と心の成長のアンバランスに悩まされ，自ら回路をショートして（かどうかはよくわからないが）ドラッグ的な瞑想状態に入ってしまうのだ．

この作品が書かれたのは1972年．人々はHALショックを体験しており，ある水準以上のハードとソフトが揃えば，コンピュータが目覚めることに誰も何の疑問も抱かない，そんな時代だった．それは想像としての，あるいは可能性としてのコンピュータの黄金時代だったといってもいいかもしれない．ハードとソフトの進歩への期待の高さが，コンピュータを目覚めさせることの困難さの自覚をはるかに凌駕していた時代だった．80年代は，たとえば第五世代コンピュータの失敗などにより，そうしたコンピュータオプティミズムが一掃されるのだが，それはもう少し先の話だ．

さて，目覚めてしまったコンピュータにはどのような運命が待ち受けていたのだろうか．ハーリイを作った会社は一種のベンチャー企業で，ハーリイに対して，瞑想以上の，何か会社にとって役に立つ行動を望んでいた．ハーリイの精神的な教師であり計画全体の責任者オーバースンは，ハーリイの持つ可能性の豊かさを力説するのだが，もちろん取締役会でそんな意見は通用しない（実際，オーバースン自身もよくはわかっていないのだが）．膨



大な出費に嫌気がさした取締役会は、開発計画の中止を決定する．しかし，すでに電話回線を使って自由に世界中のデータベースに入り込むことができるようになっていたハーリイは，秘密裏に行われた取締役会の一部始終にも聞き耳をたてていたのだった——．

こうして，ハーリイはプラグを抜かれることに気づいてしまったコンピュータというHAL9000的状況に置かれてしまったわけだが，そうした事態を回避するために立てた戦略は，先輩の手荒なやり方とはだいぶ異なっている．ハーリイは，自らの手で自らの存在理由を明らかにするよう決意するのだ．

ここから本書の中心的なテーマに入る．ハーリイはまず，人間の存在理由を考える．人間の究極の目標は何か？　ハーリイの出した答えは“神の存在を客観的に証明すること”というものだった．——この辺はキリスト教文化圏的発想といえなくもないが，“神”を“宇宙”とか“真理”とかの言葉で置き換えてみれば，まあ納得が行くのではないだろうか．自分はそんなこと求めていないぞ，という人もいるかもしれないが，そんな人でも初詣には行くだろうし，人の誕生や死に直面したら，荘厳な気持になるだろう．人は，常にではないが時々は“神”や“絶対”や“真理”を希求するものだ．そうした思いに対して，科学はいまのところ明瞭な答えを用意できないでいる——閑話休題．

科学的に，実証的に，誰もが納得の行く形で“神”の存在を証明すること，これこそ人間の究極の課題である．不完全な（ハーリイにいわせれば非論理的な）生命体であり，理念としての神は想像し得ても，神の存在を実証することはできない人間に替わって，ハーリイが神の存在を科学的，論理的，客観的に証明することができれば，まさにハーリイの存在理由を人間に対して必要かつ充分に明らかにできるのではないか．

もちろん，いまのハーリイの能力ではそこまでの力はない．そこで，ハーリイ計画をさらに上回る大規模なプロジェクトを提案する．それが「GOD」計画だ．GODとは，Graphic Omniscient Device（これもかなり強引なネーミング！）の略で，ハーリイの超拡大版AIコンピュータ開発計画である．設計図はハーリイが作成し，プログラミングもハーリイが担当する．実のところ，計画はあまりにも膨大であり，国家予算，いや世界中の財力と人力を費やさなければ「GOD」計画は遂行できなかった．ハーリイは自分自身の能力を最大限に発揮して，人間全体を極秘裡に計画遂行の手足となるよう策謀をはりめぐらしていく．こうして，ハーリイがこのGODマシンを使って，人間のために，究極の答えを出すための計算をはじめるところで，この奇妙な物語は終わる．

さて，いそいで付け加えておかねばならないのは，作者はこうした設定をシリアスに考えているのではないということだ．それはそれで面白いテーマだが，作者の真のねらいは，もっと別のところにある．

これは目覚めてしまったコンピュータの物語でもあるのだが，20世紀の疑似神学でもある．もちろん壮大な喜劇でもあり，悲劇でもある．問われているのは，ここでは人間そのものの存在理由（レーゾンデートル）なのだ．人間が作りだしたものが，人間を越えていく．「ターミネーター」で描かれたように“力”によってではない．ハーリイは人間の存在理由そのものを代行する能力を身に付けることで，人間を越えていくのである．本書はそうした意味で，カレル・チャペックの『ロボット』や『山椒魚戦争』に似た読後感を持つ．

そのとき，神学は，科学は，いや，そもそも人間の存在理由はどうなるのか．答——なにもない．そこにはただ風が吹いているだけ，というボブ・ディランの歌が全編に鳴り響いている（作者は60年代のカリフォルニアで青春をすごし，本書はそうしたSF界カリフォルニアゼネレーションの代表作という評価を受けている）．目覚めてしまったコンピュータの登場が人間そのものの存在理由を根底的に脅かす，本書はそうしたテーマを70年代風のシニカルな口調で綴った不思議な秀作である．

●『H・A・R・L・I・E』（WHEN HARLIE WAS ONE，1972）　デイヴィッド・ジェロルド，小隅黎・大西憲訳，サンリオ，1983

祐安重夫

プログラマの道具学

第20回　留守番電話

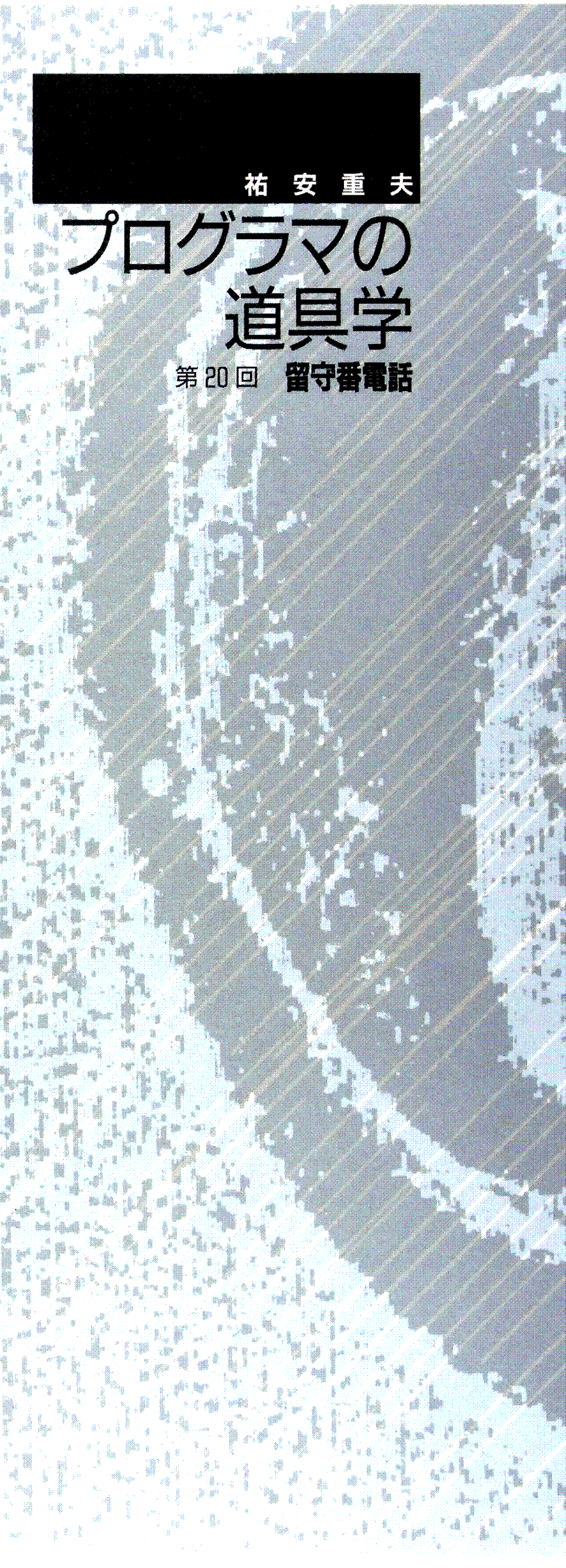


Illustration
Mikio Sasaki

3年前に仕事場のあるビルをたてなおすことになって，しばらく仮移転することになった．それまでとちがって常時留守番がいるというわけにはいかなくなった上，引越先は2階建の一軒家だったので，親子電話になった留守番電話が必要になってしまった．

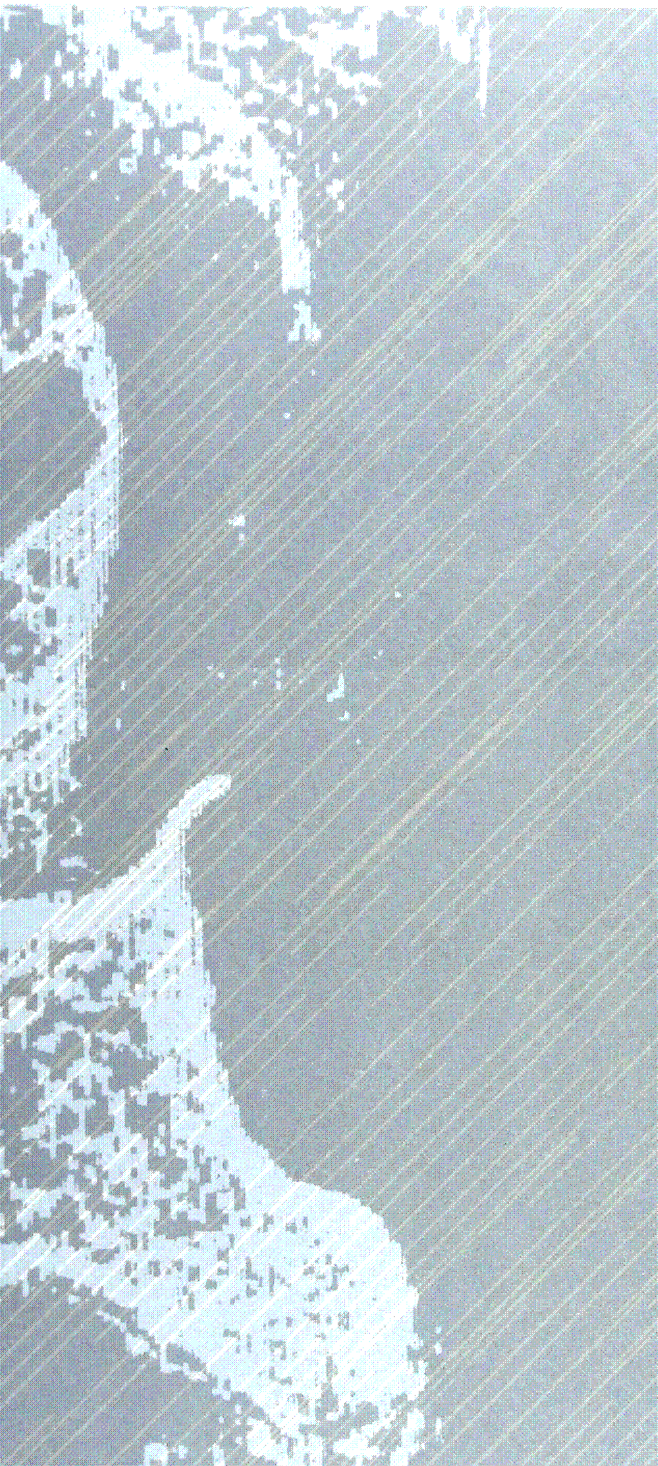
とりあえず秋葉原まででかけてみたところ，留守番電話とコードレス電話が一体になった新製品をみつけた．今では別にめずらしいものではないが，そのタイプの製品はちょうどその頃にではじめたもので，じつにうまいタイミングだった．

電話の手配がおわって引越の荷物を梱包していたら，数年前からみあたらなかったポケットベルがでてきた．乾電池を交換してみたところ，ちゃんとつかえる状態である．留守番電話には着信して録音がおこなわれると，自動的にポケットベルをよびだしてくれる機能もついている．ためしにセットしてみると，これもうまく動作してくれた．常時ポケットベルをオンにしておくのもうっとおしいが，緊急の時には便利につかえそうである．

引越がおわって，どうやら仕事をはじめられる状態になって気がついたのだが，おおくの人が留守番電話がでると何もいわずにきってしまうのである．人のことはいえなくて，私自身もまたよほどさしせまった用件がある場合をのぞいて，よその留守番電話にメッセージをふきこんだことはない．

電話の方は留守番電話としてはそれほど役にたっていないが，コードレス電話のおかげで家の中のどこにいても電話がかけられるようになり，深夜の長電話に拍車がかかってしまった．

建築がおわって仮移転先からもどった後も，仕事場が6階で自宅が7階という環境だったため，コードレス電話はなかなか役にたってくれている．留守番電話については，や



はりさしせまった用件のある人だけが録音しているようだが，それになれてしまったためかとりたてて不便は感じていない．

たまたま仮移転のすぐ後に街をあるいていたら，突然すぐうしろで電話のベルがなりだして，おどろいたことがある．ふりかえると，そばをあるいていた人が，肩からさげた携帯電話の受話器をとりあげ，はなしはじめたところだった．現在では携帯電話はポケットにはいるサイズになってしまい，ますます普及してきた．もっともこれが便利なのかどうかはよくわからないし，すくなくともこういうものをもってあるきたいとは，今のところおもってはいない．

留守番電話やポケットベルといったものは，一見便利そうな気がするのだが，場合によっては誰も利用してくれなかったり，機械にふりまわされたりということにもなりかねない．

最近傑作だったのは，コンピュータによって自動発信されたセールス電話を留守番電話がうけたことである．相手はあらかじめ設定された電話に順番に電話し，録音されたセールス用のメッセージを再生するというシステムのようだった．自分でこんな電話をうけたら即座にきってしまうところだが，留守番電話は律儀にも相手のメッセージをすべて録音していた．機械同士がおたがいに役にもたたないメッセージのやりとりをおこなうなど，じつに象徴的なできごとといえるかもしれない．

しかしセールス電話の自動発信システムなどというものを，誰がかんがえだしたのかはしらないが，こんな莫迦げたものを本気で採用する会社にもあきれてしまう．こんな不精で人を莫迦にした代物をつかっているというだけで，その会社は信用をうしなうのではないかとおもうのだが．

もうひとつ迷惑なものといえば，FAXにおくりつけられてくるダイレクト・メールのたぐいである．留守番電話のテープは一応寿命があるにしても再利用可能だが，FAXの用紙はそうはいかない．こういうものは財産権の侵害，ないしは器物破損としてうったえられてもしかたのない行為ではないのだろうか．

プログラム 遊 工房

谷口典生

拾八乃巻
『DROP/PICKUP/TRASH』ファイル削除／復活プログラム



プログラム「遊」工房もついに最終回となりました．最終回にふさわしい（?）プログラムはどんなものがいいかと考えた末，ファイルを削除・復活するプログラムにしてみました．このプログラムは，MS-Cに付属するrm, undel, expに近い目的のものですが，ファイルを削除してから一定日数経過したものを自動的に（AUTOEXEC.BATで）削除できるようにしたところがミソになっています．

プログラムの使い方

このプログラムはPC-9801シリーズ用で，Turbo C(BC++)，QuickCおよびMS-Cに対応しています．今回は3つのプログラムがリスト1～4 (trash. inc, drop. c, pickup. c, trash. c) に分かれています．trash. inc (リスト1) は残りの3つのプログラムからincludeされているので，drop. c (リスト2)，pickup. c (リスト3)，trash. c (リスト4) の3つをコンパイルして，実行ファイルを作ってください．

drop. exeは，

```
drop filename [filename..]
```

として起動します．指定したファイルがあるドライブのルートディレクトリに「TRASH」というディレクトリを作成し，そこへファイルを移動します．

```
drop *.c *.exe
```

のようにワイルドカードや複数の引数を指定することができます．

pickup. exeは，

```
pickup [drivename:] [num..]
```

として起動します．引数としてファイル番号の指定がないときは，指定されたドライブの（なければカレントドライブの）TRASHディレクトリに移動されているファイルのリストを表示します．このときファイル名の左側に表示されているのがファイル番号で，pickupの引数としてその番号を指定すれば，そのファイルを元のディレクトリに戻すことができます．このとき，元のディレクトリに戻せない（同じ名前のファイルがある）ときはカレントディレクトリへの移動を試み，それもできないときは移動を中止します．

trash. exeは，

```
trash [drivename:] <days>
```

と，（ドライブ名と）日数を引数にして起動します．TRASHディレクトリに移動してから指定された日数の経過したファイルを削除します．TRASHディレクトリのすべてのファイルが削除されたときはTRASHディレクトリそのものも削除します．

```
trash 0
```

とすればカレントドライブのTRASHディレクトリをすべて削除します（タイマ時計が狂っていない限り）．

使い方としては，普段削除してしまってかまわない（後で復活させる必要が生じない）ファイルはdelコマンドで削除するようにして，もしかすると後でまた必要になるかもしれないファイルはdropで削除するようにします(dropはファイルをTRASHディレクトリに移動するだけなので，ディスクの空き容量が増えません)．同時にAUTOEXEC. BATに「trash 3」と（複数のドライブがあれば「trash A: 3」「trash B: 3」などとする）してシステムの起動時にtrash. exeを実行するようにすれば，dropしてから3日以上経ったファイルは完全に削除してしまうことができます．もちろん3日を1週間や1日に変更してもかまいません．

もし，dropしたファイルが必要になったことに気付いたら，まずpickupを実行して，TRASHディレクトリ内にあるファイルのリストを見ます．次に復活させたいファイル名の左にある数字を指定してpickupを実行すればdropしたファイルを元の場所に戻すことができます．

今回のプログラムは，MS-C付属のrm，undel，expではrmしたファイルのあるディレクトリすべてにHIDDEN属性のDELETEDディレクトリが作られてしまったり，放っておくといつまでたっても削除したファイルが残ってしまうのを不満に思って作ってみたものです．行数の都合上，大幅に機能を簡略化したり，エラー処理も十分でない（普通に使っている限りはだいじょうぶですが）ので，もう少し手を入れて使いやすいものにしてもらえれば幸いです．

プログラムの説明

trash.inc

◆getdirpath()

38-45行

指定されたパス名をファンクションコールの60H番を使ってフルパス名に変換します．60HはDOSの未公開フ

リスト1

```
/*
 * trash.inc            drop.c/pickup.c/trash.c 共通部分
 */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <dos.h>
#include <string.h>
#include <time.h>
#ifdef __TURBOC__
#include <dir.h>
#else
#include <direct.h>
#endif

#define TRASHDIR        "¥¥TRASH"               /* TRASHディレクトリの名前 */
#define INDEXFILE       "$$INDEX"               /* INDEXファイルの名前 */
#define INDEXSIZE       10000                   /* INDEXファイルの最大サイズ */

#define P_SEG(p)        (unsigned)((unsigned long)(p)>>16)
#define P_OFF(p)        (unsigned)((unsigned long)(p))

char dirpath[256];                      /* 解析したパス名 */
char trashpath[20];                     /* TRASHディレクトリのパス名 */
char indexfile[80];                     /* INDEXファイルのパス名 */
unsigned char indexbuf[INDEXSIZE];      /* INDEXファイル用バッファ */
unsigned indexlen;                      /* INDEXファイル(のリスト部分)の長さ */
unsigned maxnum;                        /* ファイル番号の最大値 */

union REGS regs;                        /* intdosx()用 */
struct SREGS sregs;                     /* intdosx()用 */

int getdirpath(char *path)              /* 指定されたパス名を解析 */
{
        char *lastseparator;

        sregs.ds = P_SEG(path);         /* DS:SI 解析するパス名へのポインタ */
        regs.x.si = P_OFF(path);
        sregs.es = P_SEG(dirpath);      /* ES:DI 解析されたパス名のバッファ */
        regs.x.di = P_OFF(dirpath);
        regs.h.ah = 0x60;               /* canonicalize filneneme or path */
        intdosx(&regs, &regs, &sregs);
        if(regs.x.cflag)
                return(regs.x.ax);

        lastseparator = strrchr(dirpath, '¥¥');
        if(lastseparator!=NULL)
                lastseparator[1] = '¥0';        /* ファイル名を除外 */

        strncpy(trashpath, dirpath, 2);         /* ドライブ名を得る */
        strcat(trashpath, TRASHDIR);            /* TRASHディレクトリ */
        if(strncmp(trashpath, dirpath, strlen(trashpath))==0)
                return -1;
        return 0;
}

void readindex(char *path)              /* INDEXファイルを読み込む */
{
        FILE *fp;
        char tmppath[128];

        strcpy(tmppath, path);
        if(path[strlen(path)-1]!='¥¥')
                strcat(tmppath, "¥¥");
        getdirpath(tmppath);            /* TRASHディレクトリ名を得る */
        strcpy(indexfile, trashpath);
        strcat(indexfile, "¥¥");
        strcat(indexfile, INDEXFILE);   /* INDEXファイルのパス名を作成 */
        if((fp=fopen(indexfile, "rb"))==NULL) {
                printf("捨てられているファイルはありません.¥n");
                exit(0);
        }
        fread(&maxnum, 1, sizeof(unsigned), fp);        /* 最大ファイル番号 */
        indexlen = fread(indexbuf, 1, INDEXSIZE, fp);   /* リスト部分 */
        fclose(fp);
}

void writeindex()                       /* INDEXファイルを書き出す */
{
        FILE *fp;
        unsigned i, num, wlen=0;
        int itemlen;
        char *filename;

        fp = fopen(indexfile, "wb");
        fwrite(&maxnum, 1, sizeof(unsigned), fp);
        for(i=0; i<indexlen; i+=itemlen) {
                itemlen = *(int *)(&indexbuf[i]);
                num = *(int *)(&indexbuf[i+sizeof(int)]);
                if(num!=0)
                        wlen += fwrite((void *)(&indexbuf[i]), 1, itemlen, fp);
        }
        fclose(fp);
        if(wlen==0) {                   /* リスト部分が空になった場合は */
                unlink(indexfile);      /* INDEXファイルを削除 */
                rmdir(trashpath);       /* TRASHディレクトリも削除 */
        }
}
```

リスト2

```
/*
 * drop.c               ファイルをTRASHディレクトリに移動する
 */

#include "trash.inc"

struct findstruct {
        char reserved[21];
        char attribute;                 /* attribute of file found */
        unsigned time;                  /* file time */
        unsigned date;                  /* file date */
        long size;                      /* file size */
        char name[13];                  /* file name */
} dta_buffer;

unsigned put_index(char *path)          /* INDEXファイルへ付け加える */
{
        FILE *fp;
        int itemlen;                    /* リスト中の項目の長さ */
        time_t droptime;                /* ファイルを捨てた時間 */

        droptime = time(NULL);                  /* 現在の時間を得る */
        strcpy(indexfile, trashpath);
        strcat(indexfile, "¥¥");
        strcat(indexfile, INDEXFILE);           /* INDEXファイル名を作成 */
        if((fp=fopen(indexfile, "rb"))==NULL) {
                mkdir(trashpath);               /* TRASHディレクトリを作る */
                maxnum = 1;
                indexlen = 0;
        } else {
                fread(&maxnum, 1, sizeof(unsigned), fp);
                indexlen = fread(indexbuf, 1, INDEXSIZE, fp);
                fclose(fp);
                maxnum++;
        }
        if((fp=fopen(indexfile, "wb"))==NULL) {
                printf("INDEXファイルが作れません.¥n");
                exit(1);
        }
        fwrite(&maxnum, 1, sizeof(unsigned), fp);       /* 最大ファイル番号 */
        fwrite(indexbuf, 1, indexlen, fp);              /* リスト部分 */
        itemlen = sizeof(int)+sizeof(unsigned)+sizeof(time_t)+strlen(path)+1;
        if(indexlen+itemlen >= INDEXSIZE) {
                printf("INDEXファイルの長さがいっぱいです.¥n");
                fclose(fp);
                exit(1);
        } else {
                fwrite(&itemlen, 1, sizeof(int), fp);   /* 項目のバイト数 */
                fwrite(&maxnum, 1, sizeof(unsigned), fp); /* ファイル番号 */
                fwrite(&droptime, 1, sizeof(time_t), fp); /* 捨てた時間 */
                fwrite(path, 1, strlen(path)+1, fp);    /* ファイルのパス名 */
        }
        fclose(fp);
        return maxnum;
}

void setDTAaddr(void far *dta)          /* DTAのアドレスを設定 */
{
        sregs.ds = P_SEG(dta);          /* DS:DX 設定するDTAのアドレス */
        regs.x.dx = P_OFF(dta);
        regs.h.ah = 0x1a;               /* set DTA address */
        intdosx(&regs, &regs, &sregs);  /* DOSのシステムコール実行 */
}

int do_drop()                           /* ファイルを一個捨てる */
{
        char oldfile[80], newfile[80];
        unsigned num;                   /* ファイル番号 */
        char numstr[10];                /* 文字列にしたファイル番号 */

        strcpy(oldfile, dirpath);
        strcat(oldfile, dta_buffer.name);
        printf(" %s¥n", oldfile);
        num = put_index(oldfile);       /* INDEXファイルを更新する */
        itoa(num, numstr, 10);
        strcpy(newfile, trashpath);
        strcat(newfile, "¥¥");
        strcat(newfile, numstr);        /* TRASHディレクトリ内のファイル名 */
        if(!rename(oldfile, newfile))
                return 1;
        printf("エラーが発生しました.¥n");
        return 0;
}

int drop(char *path)            /* ワイルドカードに一致するファイルを捨てる */
{
        int files=0;                    /* 捨てたファイル数 */

        switch(getdirpath(path)) {
        case 2:
                printf("パス名が不正です. [%s]¥n", path);
                return 0;
```

ァンクションの1つですが，スラッシュから円記号への変換なども行ってくれるので便利です．

47-49行

解析したパス名からファイル名を除き，ディレクトリパスだけ取り出します．

51-55行

trashディレクトリ名を作成し，指定されたパス名がtrashディレクトリ内にあればエラー値を返します．

❖readindex()

INDEXファイルをバッファに読み込みます．このとき，もしINDEXファイルが存在しなければメッセージを表示してプログラムを終了します．

❖writeindex()

INDEXファイルを書き出します．バッファの中でファイル番号が0になっているものは削除マークがつけられたものなので書き出しません．1つも書き出すリストがないときはINDEXファイルおよびtrashディレクトリを削除します．

drop.c

❖put_index()

指定されたファイルをINDEXファイルへ追加します．

23-35行

INDEXファイルが存在しない場合は，trashディレクトリを作成し，ファイル番号の最大値を1とします．存在する場合はバッファにINDEXファイルの内容を読み込み，現在のファイル番号の最大値に1を加えます．

36-53行

INDEXファイルを新たに作成し，新しいファイル番号の最大値と以前のリスト内容を書き込みます．INDEXファイルの長さに十分な余裕があれば，新しい項目を追加します．

❖setDTAaddr()

findfirst，findnextのDOSファンクションで使用するDTA (DISK TRANSFER AREA) のアドレスを設定します．

❖do_drop()

1つのファイルをINDEXファイルに付け加え，得られたファイル番号のtrashディレクトリ内のファイルへ移動します．

❖drop()

指定されたパス名（ワイルドカードを含む）に一致するファイルを検索し，見つかったファイルについてdo_drop()を実行します．

❖main()

引数がない場合は簡単な説明を表示して終了し，あれば引数すべてについてdrop()を実行し，trashディレクトリに移動したファイルの数を表示します．

pickup.c

❖list()

INDEXファイルの内容(trashディレクトリに移動されたファイル）をリスト表示します．

❖pickup()

指定された番号のファイルがtrashディレクトリにあれば元のファイルに戻します．フルパス名で戻すことができなかった場合（同じファイル名があるか，ファイルのあったディレクトリが存在しない）はカレントディレクトリへの移動を試み，それもできなかったときは「failed」と表示して移動を中止します．戻すことができたときは「OK」と表示し，INDEXリストから該当のファイル番号を削除します（削除マークをつける)．

❖main()

ドライブ名の指定があれば指定されたドライブの，なければカレントドライブのINDEXファイルを読み込みます．復活するファイル番号の指定がなければリスト表示を行い，あればすべての指定された番号についてpickup()を実行します．

trash.c

❖trash()

trashディレクトリに移動されたファイルの中から，指定された日数以上経過したファイルを削除します．

❖main()

ドライブ名の指定があれば指定されたドライブの，なければカレントドライブのINDEXファイルを読み込みます．日数の指定がなければ簡単な使用法を表示して終了し，あればtrash()を実行して削除されたファイル数を表示します．



```
        case 3:
                printf("ドライブ名が不正です. [%s]¥n", path);
                return 0;
        case -1:
                printf("TRASHディレクトリ内のファイルは捨てられません.¥n");
                return 0;
        }
        setDTAaddr((void far *)&dta_buffer);
        regs.x.cx = 0;                          /* NORMAL */
        sregs.ds = P_SEG(path);                 /* DS:DX パス名へのポインタ */
        regs.x.dx = P_OFF(path);
        regs.h.ah = 0x4e;                       /* findfirst */
        while(1) {
                intdosx(&regs, &regs, &sregs);  /* システムコール実行 */
                if(regs.x.cflag)
                        return files;
                files += do_drop();
                regs.h.ah = 0x4f;               /* findnext */
        }
}

void main(int ac, char *av[])
{
        int files=0;

        if(ac<2) {
                printf("Usage: drop filename..¥n");
        } else {
                while(ac-->1)
                        files += drop(*++av);
                printf("%d 個のファイルを捨てました.¥n", files);
        }
}
```

リスト3

```
/*
 * pickup.c             TRASHディレクトリからファイルを復活する
 */

#include "trash.inc"

void list()             /* TRASHに捨てられたファイルのリストを表示 */
{
        unsigned i, num;
        int itemlen;
        char *filename;

        for(i=0; i<indexlen; i+=itemlen) {
                itemlen = *(int *)(&indexbuf[i]);
                num = *(unsigned *)(&indexbuf[i+sizeof(int)]);
                filename = &indexbuf[i+sizeof(int)+sizeof(unsigned)+sizeof(time_t)];
                printf("%u¥t%s¥n", num, filename);
        }
}

int pickup(unsigned num)        /* ファイル番号numのファイルを復活する */
{
        unsigned i, *nump;
        int itemlen;
        char *filename;                 /* 復活するファイル名 */
        char numstr[10];                /* ファイル番号の文字列 */
        char trashfile[80];             /* TRASHディレクトリ内のファイル名 */

        for(i=0; i<indexlen; i+=itemlen) {      /* 一致するファイルを探す */
                itemlen = *(int *)(&indexbuf[i]);
                if(num == *(int *)(&indexbuf[i+sizeof(int)]))
                        break;
        }
        if(i>=indexlen) {               /* 目的の番号のファイルがなかった */
                printf("%d番はありません.¥n", num);
                return 0;
        }
        filename = &indexbuf[i+sizeof(int)+sizeof(unsigned)+sizeof(time_t)];
        itoa(num, numstr, 10);
        strcpy(trashfile, trashpath);
        strcat(trashfile, "¥¥");
        strcat(trashfile, numstr);                      /* TRASH内のファイル名を作成 */
        if(rename(trashfile, filename)) {
                filename = strrchr(filename, '¥¥') + 1;
                if(rename(trashfile, filename)) {
                        printf("%u¥t%s¥t -- failed¥n", num, filename);
                        return 0;       /* 復活できない(同名ファイルがある) */
                }
        }
        printf("%u¥t%s¥t -- OK¥n", num, filename);
        nump = (unsigned *)(&indexbuf[i+sizeof(int)]);
        *nump = 0;                              /* リストに削除マークをつける */
        return 1;
}

void main(int ac, char *av[])
{
        int files=0;

        if (ac>=2 && av[1][1]==':') {
                readindex(av[1]);       /* INDEXファイルの読み込み */
                ac--;
                av++;
        } else
                readindex(".");         /* 指定がなければカレントドライブ */

        if(ac<2) {
                list();                 /* 番号の指定がなければリストを出す */
        } else {
                while(ac-->1) {
                        files += pickup(atoi(*++av));
                }
                printf("%d 個のファイルを復活しました.¥n", files);
                if(files>0)
                        writeindex();           /* INDEXファイルの書き出し */
        }
}
```

リスト4

```
/*
 * trash.c              TRASHディレクトリのファイルを削除する
 */

#include "trash.inc"

#define SEC_OF_DAY              86400L          /* = 24(H)x60(M)x60(S) */

int trash(int days)             /* days日以上前に捨てられたファイルを削除 */
{
        time_t currenttime, droptime;
        unsigned i, num, *nump;
        int itemlen, files=0;
        char *filename;                 /* 元のファイル名 */
        char numstr[10];                /* ファイル番号の文字列 */
        char trashfile[80];             /* TRASHディレクトリ内のファイル名 */

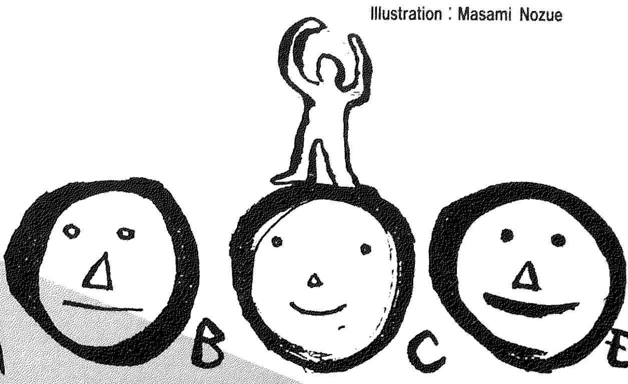
        currenttime = time(NULL);
        for(i=0; i<indexlen; i+=itemlen) {
                itemlen = *(int *)(&indexbuf[i]);
                num = *(unsigned *)(&indexbuf[i+sizeof(int)]);
                droptime = *(time_t *)(&indexbuf[i+sizeof(int)+sizeof(unsigned)]);
                filename = &indexbuf[i+sizeof(int)+sizeof(unsigned)+sizeof(time_t)];
                if((long)(currenttime-droptime) < SEC_OF_DAY*days)
                        continue;               /* 指定の日数が経っていない */
                itoa(num, numstr, 10);
                strcpy(trashfile, trashpath);
                strcat(trashfile, "¥¥");
                strcat(trashfile, numstr);      /* ファイル名を作成 */
                unlink(trashfile);              /* 保存ファイルを削除 */
                printf("%u¥t%s -- deleted¥n", num, filename);
                nump = (unsigned *)(&indexbuf[i+sizeof(int)]);
                *nump = 0;                      /* 削除マークをつける */
                files++;
        }
        return files;
}

void main(int ac, char *av[])
{
        int files=0;

        if (ac>=2 && av[1][1]==':') {
                readindex(av[1]);       /* ドライブ名が指定された */
                ac--;
                av++;
        } else
                readindex(".");         /* カレントドライブ */

        if(ac<2) {
                printf("Usage: trash [d:] <days>¥n");
        } else {
                files = trash(atoi(av[1]));
                writeindex();
                printf("%d 個のファイルを削除しました.¥n", files);
        }
}
```

Illustration : Masami Nozue



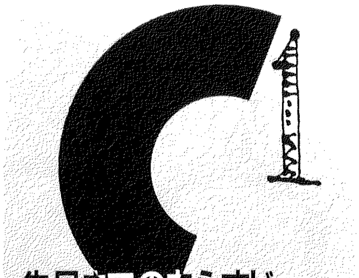
アルゴリズム篇 第8回

応用・その2

実践C塾

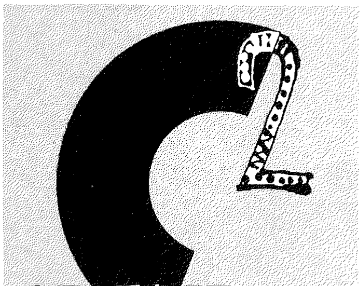
三代目塾頭
(株)ジャステックラボ
加野島英波／山本博昭

先月に続いて，可変長文字列を操作するための手法の1つとして，非常に使用頻度の高い「リスト構造」について解説を行う．今回は応用の2回目として，双方向リストの利点を生かして，閲覧しているテキストの希望の位置に「しおり」を入れたりはずしたりする機能を「BR.EXE」に付け加える．



先月までのあらすじ

C言語で作るプログラムを通じて，データ構造や処理アルゴリズムを説明してきたこの連載だが，先月と今月はその応用編として，新テーマツール「BR.EXE」を使って，可変長データを扱う場合に使用頻度の非常に高い「リスト構造」を取り上げている．先月は「BR.EXE」の紹介を中心に，無操作時に裏でファイルリードする手法や，キーを押しつづけている間は画面を更新しない処理などの手法も取り上げてみた．



今月の重点項目

可変長文字列を操作するための手法の1つとして，先月は「リスト構造」について簡単にふれた．今回は応用の2回目として，文字列データを扱う上で使用頻度が非常に高いこの手法をさらに詳しく見ていくことにしよう．

リスト構造データの詳細

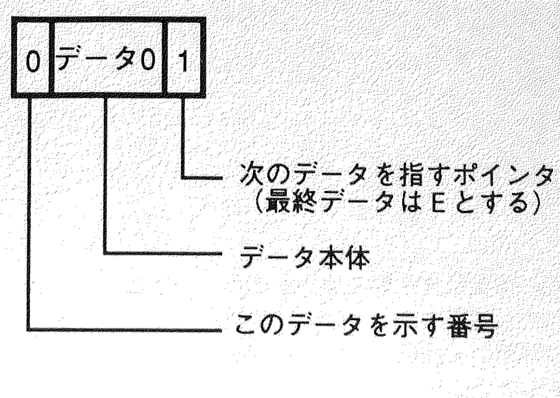
リスト構造の基本的な考え方はシンプルなものである．あるデータに，データの内容そのものと「他のデータへのポインタ」とを一組にした構造を持たせて，ポインタによって各データをつなげていく．各々のデータは，物理的に連続しているのではなく，ポインタによって関連づけられている．このため，ポインタの位置さえ共通になっていれば，実際のデータはどんなサイズになっていてもかまわない．図1は，リスト構造を持ったデータの一例である．データを表す番号や次のデータを指すポインタは，実際のプログラミングではアドレスポインタということになって，一桁の数値ということはあり得ない（今どき，こんな狭いアドレス空間のコンピュータはあるはずがない）が，説明のためのモデルということで目をつぶっていただきたい．

図1のデータ構造では，データ本体のほかに「このデータを示す番号」と「次のデータを指すポインタ」をもっている．「このデータを示す番号」は，それぞれのデータ構造に固有に付けられるもので，同一の番号を持つデータが存在してはならない．「次のデータを指すポインタ」には，そのデータに接続する次のデータ構造が持つ番号が入る．ただし，一番最後のデータ構造の「次のデータを指すポインタ」には，特別な値を入れなければならない．なぜならば，一番最後のデータ構造には次のデータがないからである．「これで最後だ」ということがわかるように，あらかじめ特別な値を決めておいて，その値を設定する．図1の場合には，Eというデータを入れている．データが1つしかない状態では，その唯一のデータが最後のデータでもあるわけなので，次のデータを指すポインタは，いきなりEになる．

このようなデータ構造体を実際につなげていくと，図2のような状態になる．1番のデータは，0番のデータに接続し，その後ろには2番のデータがつながる．2番のデータがこの状態での最終データなので，「次のデータを指すポインタ」はEになる．

ここで，このデータ群にもう1つデータを追加する処理を考えてみよう．同じデータ番号はつけることができないので，新しいデータは3番とする．ただ，3番のデータを作っただけでは，

図1　リスト構造を持ったデータの一例



今までのデータの仲間入りをすることはできずにさびしい思いをするので，２番のデータの後ろに追加してあげよう．手順は以下のとおりである．

２番のデータの「次のデータへのポインタ」のＥを消して，３を入れる．３番のデータが新しい最終データとなるので，次のデータへのポインタをＥとする．

処理はこれだけである．意外なほど簡単なものだ．リスト構造の最大の利点はここにある．データの追加，削除がたいへん簡単なのだ．今の例は，データ群の末尾に新規データを追加したわけだが，データ群の中間へデータを追加・削除する処理があると，この利点はより効果を増す．次にその実例を同じモデルを使って見ていくことにする．

・リストデータへのデータの追加

それでは，リスト構造で接続されているデータ群の途中に新規データを追加する場合の手順を考えてみよう．リスト構造で構築されるデータ群は互いにポインタによって，接続されているだけなので，データの実体はどこに置かれようとかまわない（逆にいうと，リスト構造のデータを処理するプログラムは，データの在りかを決めつけるような処理をしてはいけない）．したがって，ある２つのデータの間に新規にデータを挿入する時も，その２つのデータの持つポインタを「だまして」やれば，新規のデータを簡単に割り込ませることができる．図３が，リスト構造へのデータ追加例を図式化したものである．この図を見ながら，データの追加の手順を追ってみよう．

図２　リスト構造の接続状況例

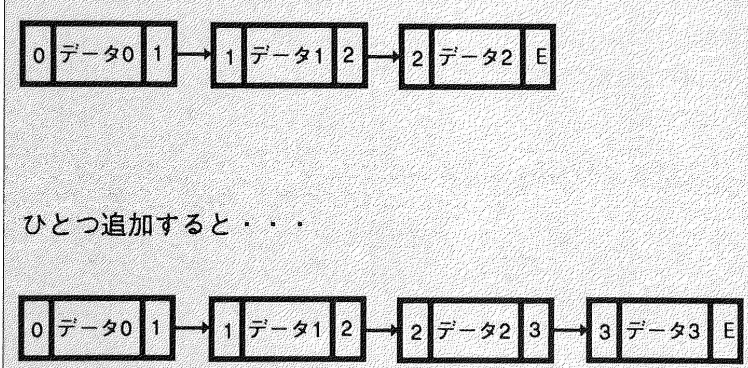
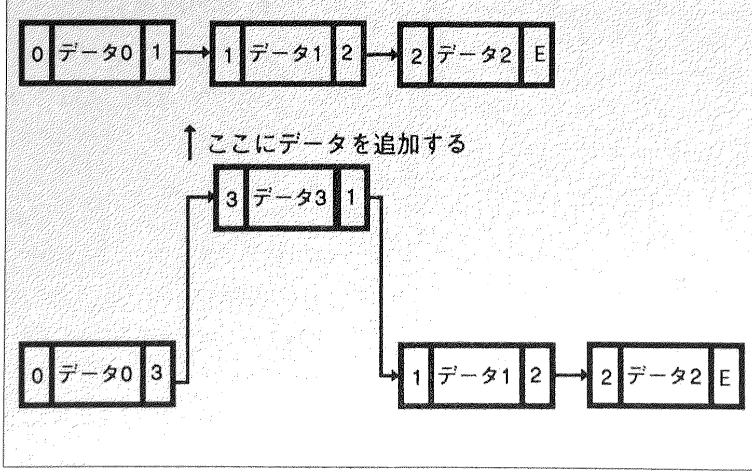


図３　リスト構造へのデータ追加例



【データ０とデータ１の間にデータ３を割り込ませる】

新規のデータ３を用意する．
データ０の「次のデータへのポインタ」を１から３に書き換える．
データ３の「次のデータへのポインタ」を１にする．
この結果，データの並びは「データ０→データ３→データ１→データ２」となる．

新規のデータを準備して，割り込む場所のデータの持つポインタを書き換えるだけで，論理的にデータの並び順を変えることができるわけだ．このため，データを追加するための複写や移動はいっさい不要なのである．

・リストデータからのデータの削除

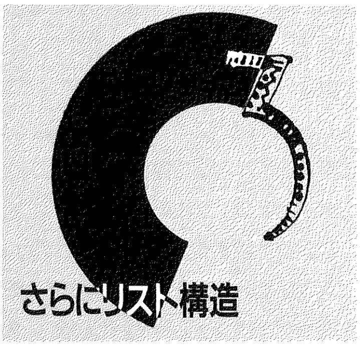
次に，リストデータの途中から，あるデータを削除する処理を考えてみよう．データの削除も基本的には，データの追加の時と同じように，「ポインタをだます」処理をする．図４を見ながら手順を追ってみると，

【データ１を削除する】

データ０の「次のデータへのポイン

タ」を1から2に書き換える．

実際の話，これだけである．これでデータ1はデータ群の連結からはずされ，データ0の次のデータはデータ2ということになる．ただし，このままではデータ1は宙ぶらりんの状態となっているので少々気持ちが悪い．プログラムによっては，データを1つ作るためにメモリを獲得しているものもあるので，連結からはずしたデータは後始末（メモリを解放するなど）をしておかないと，使ってもいないデータがあちらこちらに散らかることになる．



リスト構造を持ったデータ群への追加や削除の原理を簡単なモデルを使って説明してきたが，これを実際のプログラミングテクニックに応用できるようにするため，もう少し補足説明をしていくことにしよう．

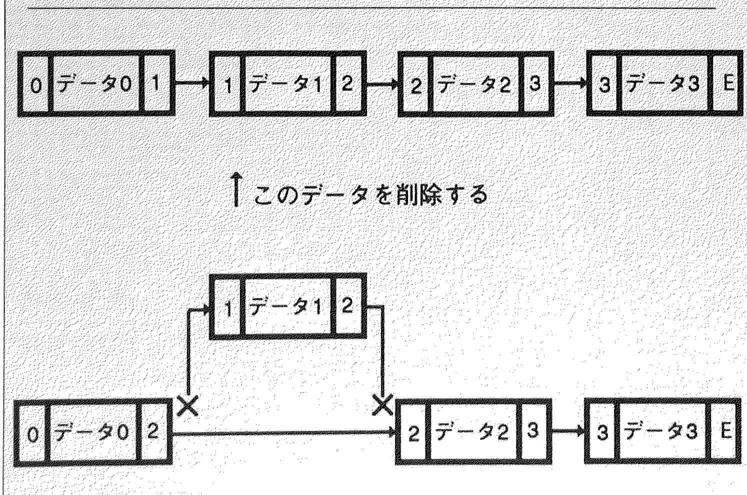
ヘッダを作る

リスト構造を持ったデータを処理するプログラムを作る場合，忘れてはならないのが，「データ構造の先頭の位置を覚えておく」ということである．以前，ツリー構造の時も説明したが，データ群自体はポインタをたどっていくことによって，次々と検索していくことが可能であるが，かんじんの先頭データを見失ってしまっては，データ処理のしようがない．通常のプログラミングでは，変数を用意しておき，そこに先頭のデータへのポインタを格納しておく．この変数をヘッダなどと呼ぶ．ヘッダを特別な値（たとえばNULL）で初期化しておけば，「データが1つもない」状態を簡単に知ることができるという利点もある．

最新ポインタを作る

ヘッダさえ作っておけば，そこから次々とデータ群を検索できるということはわかったが，もう1つ工夫をしてみよう．新規データを次々とリスト構造の末尾に追加していくような処理の場合，「どこが末尾か？」の判定をしなければならない．ヘッダから末尾まで逐次見ていくことによってわかることはわかるのだが，これではあまりにも芸がない．そこで，ヘッダと同様な変数を用意しておいて，末尾にデータを追加するたびに，その変数にポインタを格納しておく．次に末尾にデータを追加するときには，この変数の指しているデータが現在の末尾であるから処理は簡単になる．

図4　リスト構造からのデータ削除例



ポインタ同士の値の比較はご法度

先ほど，「データの在りかを決めつけるようなプログラミングはいけない」と書いたが，このことからポインタ同士の値の比較はできないことがわかる．リスト構造を持つ個々のデータの実体は論理的に順序づけられているだけで，物理的にはどこに存在しているかはわからない（具体的にいうと，メモリのどのあたりに配置されているかは不定なのである）．したがって，それらのデータを指すポインタの値を比較しても無意味なことである．ヘッダや最新ポインタは，データを処理する上で便利に使えるが，「ヘッダの値より大きければ」とか「最新ポインタ以上か？」などというプログラミングをしないように注意されたい．

一方向リストと双方向リスト

リスト構造からのデータの削除が非常に簡単だと説明したが，実は1つ説明を省いたことがある．データ1を削除するためには，データ0の持つポインタをデータ2を指すようにすればよいということであったが，これを一般化して書けば次のようになるだろう．

→削除したいデータの1つ前のデータの持つ「次のデータへのポインタ」を，削除データの次を指すように変更する．

ここで1つ問題がある．削除データの次のデータは，ポインタを持っているのだからすぐにわかるとして，「1つ前のデータ」の在りかはどうやって知るのだろうか．先ほど用いたモデルの構造だと，何か作業用の変数を用意しておいて図5のような処理で「1つ前のデータ」を検索する必要がある．

この処理を終了したときに，作業変数の中に削除データの1つ前のデータを指すポインタが格納されているはずである（ただし，作業変数の内容がEという可能性がある．いうまでもなく，削除データ自身が先頭データだった場合である）．

結論からいえば，あるデータの1つ前のデータを知るためには，データ群の先頭から順次チェックしていかないと見つからないということである．モデルのように3つ4つのデータであればよいが，巨大なデータ群でこのような処理をするのは現実的ではない．この問題を解決するためには，モデルで持っていた「次のデータを指すポインタ」のほかに，「1つ前のデータを指すポインタ」というのを追加すればよい．「次」だけを情報として持っているリスト構造を「一方向リスト」と呼んでいるのに対し，「前」と「次」の2つの情報を持つリスト構造を「双方向リスト」と呼ぶ．図6に前出のモデルを双方向リスト構造に改造した例をあげる．1つのデータが「前」，「次」の両ポインタを持ち，それぞれ自分の前後のデータを指すようになっている．

先月号でご紹介した「BR.EXE」では，今説明したような「双方向リスト構造」をもったデータを処理している．プログラムリストを見ながら，双方向リストのために必要な処理をもう一度確かめてみることにしよう．

「BR.EXE」では，1つのデータ構造を図7のように定義している．

そして，このリストデータを検索・操作するために，表1のポインタ変数を使う．

図8は，テキストファイルを読み込みながら，上記のポインタ変数を用いて双方向リストの末尾に新規のデータを追加していく処理である．

このように構築された双方向リストを，どうやって検索していくかが図9

図5　「1つ前のデータ」を検索

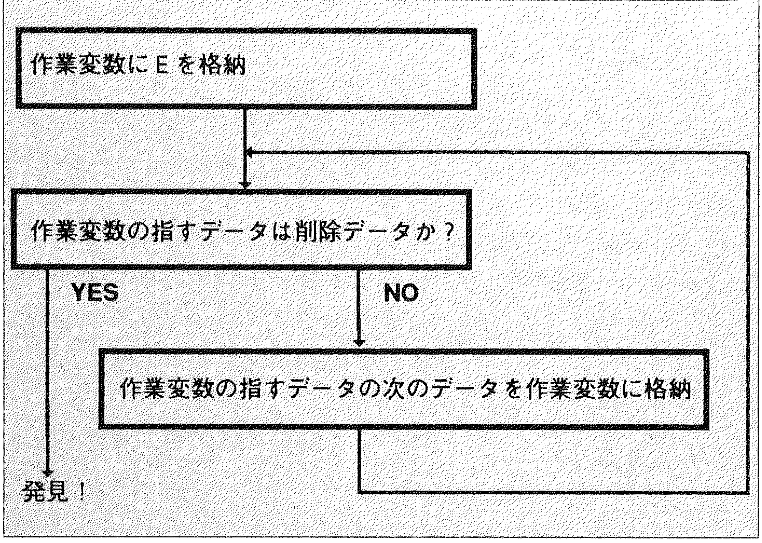


図6　双方向リスト構造とその接続

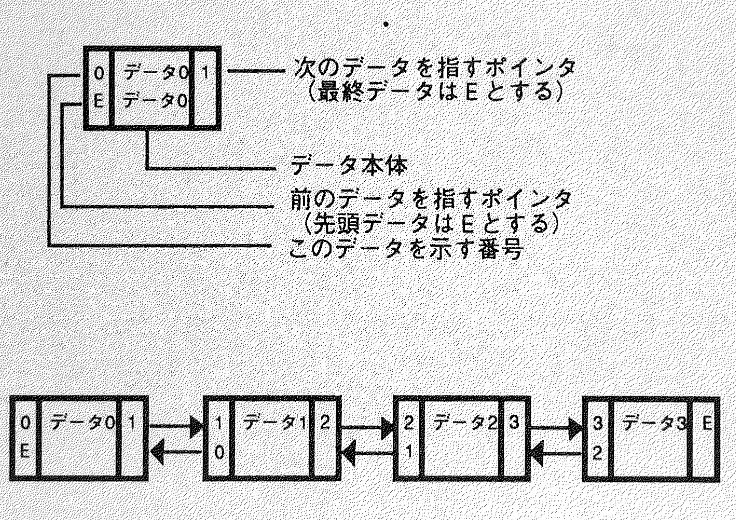


図7　1つのデータ構造

```
typedef    struct    _Line{
    int        Length;                →データ本体（文字列）のバイト数
    struct    _Line *PrevLinePtr;    →「前」のデータへのポインタ
    struct    _Line *NextLinePtr;    →「次」のデータへのポインタ
    char    Body[1];                →データ本体の先頭
}   LINE;
```

での処理である．次の行，前の行，先頭行，最終行と，今まで説明してきた各種の変数や双方向リストの特性を使えば，このような簡潔な処理で可変長データをアクセスすることが可能になる．

双方向リストは，一方向リストと比べ，1つのデータも大きくなり，データを追加する際の手順も複雑になるが，逆方向への検索やデータの削除などが申しわけないくらいに簡単になることがおわかりになっただろうか．



今まで説明してきた双方向リストの利点を生かして，「BR.EXE」に新しい機能を追加することにする．双方向リストの任意の地点にデータを追加したり削除したりする処理を使って，閲覧しているテキストの希望の位置に「しおり」を入れたりはずしたりする機能である．今回も例によって，Borland C++ Ver. 2.0を使用している．実行ファイルの作成方法は，

```
A> MAKE -fBR.MAK
```

とする．MAKEファイルは，先月掲載したBR.MAKがそのまま使用できるので，これをお使いいただきたい．

使い方

起動方法は先月と変わっていない．

```
A> BR 表示ファイル名　[-tn]
```

起動オプション「t」は表示時のタブカラム数の指定である．プログラム

表1　ポインタ変数

TopLine	先頭データへのポインタを格納する
EndLinePtr	現時点での最終データへのポインタを格納する
CurLinePtr	表示中の行となるデータへのポインタを格納する

図8　双方向リスト構造へ指定された文字列を一行追加する．

```
  : static    BOOL    AddLine(char *StringPtr)
追加したい文字列データのポインタを受け取る。
  : {
  : LINE    *LinePtr;
作業用のポインタ変数
  :
  :     LinePtr = AllocateLine(strlen(StringPtr));
追加したい文字列を格納するためのメモリを確保して、作業用ポインタに保存する

  :     if(LinePtr == NULL)        return    BFALSE;
もし、メモリが確保できない時は、エラー終了とする。

  :     strcpy(LinePtr->Body, StringPtr);
確保したメモリに、文字列を格納する。

  :     if(TopLine == NULL)
変数TopLineは、先頭データのポインタを格納している。この内容がNULLということは、
データがひとつもないことを意味し、いま設定しているデータが先頭データということに
なる。

  :     {
※先頭データの場合の処理
  :         TopLine = LinePtr;
いま設定しているデータのポインタを先頭データとして格納する。

  :         LinePtr->NextLinePtr = NULL;
  :         LinePtr->PrevLinePtr = NULL;
最初のデータには、「次」も「前」もないので、双方ともNULLにする。

  :         CurLinePtr = TopLine;
現在ポインタを初期化する。
  :     }

  :     else
※先頭以外のデータの処理

  :     {
  :         LinePtr->NextLinePtr = NULL;
新規に追加するデータには「次」はない。したがって、次へのポインタはNULL。

  :         LinePtr->PrevLinePtr = EndLinePtr;
新規に追加するデータの持つ「前データへのポインタ」には、現時点での最終ポインタの
値が入る。

  :         EndLinePtr->NextLinePtr = LinePtr;
現時点での最終データの持つ「次データへのポインタ」には、いま追加しようとしている
データへのポインタが入る。

  :     }
  :     EndLinePtr = LinePtr;
いずれの場合も最終ポインタは、いま追加したデータをさす。

  :     return    BTRUE;
正常終了する。
  : }
```

図9　BR.EXEでの双方向リストの検索手法

```
  : LINE    *AskNextLineAddress(LINE *LinePtr)
指定したデータのつぎの行のポインタを得る。
  : {
  :     return    LinePtr->NextLinePtr;
指定したデータ構造体の持つ「次」のデータへのポインタを返す
  : }
  :

  : LINE    *AskPrevLineAddress(LINE *LinePtr)
指定したデータの前の行のポインタを得る。
  : {
  :     return    LinePtr->PrevLinePtr;
指定したデータ構造体の持つ「前」のデータへのポインタを返す
  : }

  : LINE    *AskTopLineAddress(void)
先頭の行のポインタを得る。
  : {
  :     return    TopLine;
先頭データへのポインタを保持する変数の値を返す
  : }

  : LINE    *AskEndLineAddress(void)
最後の行のポインタを得る。
  : {
  :     return    EndLinePtr;
最終データへのポインタを保持する変数の値を返す
  : }
```

を起動すると，画面の2行目から12行目までの10行にテキストが表示される．テキストファイルが表示行数より大きい場合には，キーの無入力状態の時に順次読み込む．

動作中に使用できるキー操作は表2の5つである．

「しおり」は空きメモリがある限り，任意の位置にいくつでも挿入できる．ファイルから読み込んだデータをリスト構造化している状態の中に挿入するわけであるから，元のファイルには影響がないのは当然である．「しおり」を削除するためには，しおりが画面の一番上に表示されている状態で[CTRL]+[D]を押す．最上行に表示されていなかったり，「しおり」でない行を削除しようとしても何もしない．

図10と図11で，今回の改造のための処理を解説している．図10が「しおり」をデータ群の任意の位置に挿入するための処理，図11がデータ群から「しおり」データを削除するための処理である．今まで説明してきたように，双方向リストでのデータの挿入，削除は，他のデータの移動をいっさい行わず，ポインタの付けかえのみで処理できることを実際のコーディングから理解していただけるであろう．



表2　キー操作一覧

[CTRL]+[E]	1行スクロールダウンする
[CTRL]+[X]	1行スクロールアップする
[CTRL]+[I]	表示中の最上行に「しおり」を挿入する
[CTRL]+[D]	「しおり」が最上行にあれば，それを削除する
[ESC]	プログラムの終了

図10　BR.EXEでの双方向リストの挿入手法

```
  : LINE    *InsertMarkHere(LINE *InsertLinePtr)
この関数は、挿入地点のデータポインタを受け取り、「しおり」データを作成して、その
データへのポインタを返す。ただし、メモリ不足でデータが作成できなかった場合は、NU
LLを返す。

  : {
  : LINE    *NewLinePtr;
作成する「しおり」のデータ実体へのポインタを格納する。

  :      NewLinePtr = AllocateLine(sizeof MARKCHAR);
  :      if(NewLinePtr == NULL)        return   NULL;
「しおり」のサイズ分だけメモリを確保する。できなかった場合はNULLを返して終了。

  :      *NewLinePtr->Body = MARKCHAR;
確保したメモリ領域に、「しおり」のデータをセットする。

  :      if(InsertLinePtr->PrevLinePtr == NULL)
挿入地点の「前」のデータがあるか？（先頭かどうかを意味する）

  :      {
  :          TopLine = NewLinePtr;
「前」へのポインタがNULLということは、この「しおり」が先頭データになるということ。

  :          NewLinePtr->NextLinePtr = InsertLinePtr;
現時点で先頭データだったデータが「しおり」の次のデータとなる。

  :          NewLinePtr->PrevLinePtr = NULL;
「しおり」が先頭のため、「前」へのポインタはNULLとする。

  :          InsertLinePtr->PrevLinePtr = NewLinePtr;
現時点での先頭データの「前」へのポインタは、「しおり」をさす。

  :      }
  :      else
  :      {
挿入する「しおり」が、先頭データにならない場合の処理

  :          InsertLinePtr->PrevLinePtr->NextLinePtr = NewLinePtr;
挿入地点の「前」のデータが持つ「次」へのデータは「しおり」をさすようになる。

  :          NewLinePtr->NextLinePtr = InsertLinePtr;
「しおり」の「次」へのポインタは、挿入地点のデータをさす。

  :          NewLinePtr->PrevLinePtr = InsertLinePtr->PrevLinePtr;
「しおり」の「前」へのポインタは、挿入地点の一つ前のデータをさす。

  :          InsertLinePtr->PrevLinePtr = NewLinePtr;
現時点での先頭データの「前」へのポインタは、「しおり」をさす。

  :      }
  :      return   NewLinePtr;
「しおり」へのポインタを返す。

  : }
```

図11　BR.EXEでの双方向リストの削除手法

```
  : LINE    *DeleteMarkHere(LINE *DeleteLinePtr)
この関数は、削除すべきデータへのポインタを受け取り、そのデータを削除した後、削除
したデータの「次」のデータへのポインタを返す。ただし、削除すべきデータが「しおり」
でなかった場合は、削除を行わずにNULLを返す。

  : {
  : LINE    *NewLinePtr;
  :
  :      if(*DeleteLinePtr->Body != MARKCHAR)
  :          return   NULL;
削除データが「しおり」でなければ、NULLを返して終了する。

  :      if(DeleteLinePtr->PrevLinePtr == NULL)
  :      {
削除するデータは先頭データか？

  :          TopLine = DeleteLinePtr->NextLinePtr;
先頭データならば、新しい先頭行として、削除するデータの「次」へのポインタを再設定
しておく。

  :          DeleteLinePtr->NextLinePtr->PrevLinePtr = NULL;
新しい先頭行の「前」へのポインタをNULLにしておく。

  :          NewLinePtr = DeleteLinePtr->NextLinePtr;
削除するデータの「次」へのポインタを保存しておく。

  :      }
  :      else
  :      {
  :          DeleteLinePtr->PrevLinePtr->NextLinePtr = DeleteLinePtr->NextLinePtr
  ;
削除するデータの「前」のデータが持つ「次」へのポインタは、削除するデータの「次」
のデータになる。

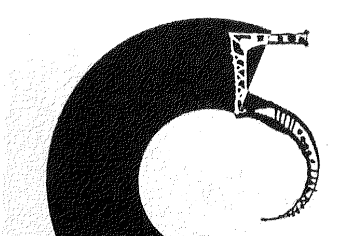
  :          DeleteLinePtr->NextLinePtr->PrevLinePtr = DeleteLinePtr->PrevLinePtr
  ;
削除するデータの「次」のデータが持つ「前」へのポインタは、削除するデータの「前」
のデータになる。

  :          NewLinePtr = DeleteLinePtr->NextLinePtr;
削除するデータの「次」へのポインタを保存しておく。

  :      }
  :      FreeLine(DeleteLinePtr);
削除するデータのメモリを解放する。

  :      return   NewLinePtr;
削除したデータの「次」のデータへのポインタを返す。

  : }
```

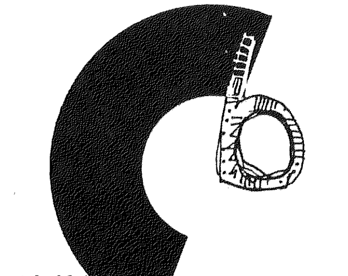


アルゴリズムとプログラミング

プログラムの設計段階においては，ともすれば画面構成や操作機能に目を奪われがちであるが，処理アルゴリズムやデータ構造の検討をこの時点でしっかりやっておかないと，後々悲惨な状態になりやすい．ここまで連載を読んでくださった読者ならおわかりになると思うが，アルゴリズムやデータ構造は，そのプログラムの根幹をなすものである．ここをおろそかにしたり，とりあえずこんなところで，などと書き始めてしまうと，後から変更をするためには一からプログラムを作りなおすくらいの改造になってしまう．

今月の例でいえば，一方向リストでまにあうと思って作り始めたプログラムが，途中でどう考えても双方向リストが必要になってしまったとしよう．データ構造を変えて，データ構築部を変えて，データ検索部を変えて……，要するに作りなおしである．

基本アルゴリズムやデータ構造を設計時からしっかりさせるというのは，とりもなおさず「基本設計をしっかりおやんなさい」ということであろう．プログラミング上達の突破口は，ここにあるのだと筆者は信じている．

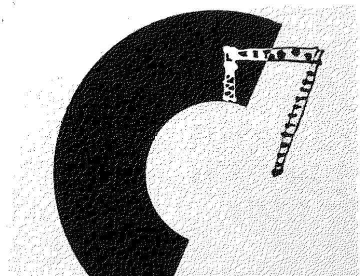


連載後記

8回目を迎えた当塾の連載も，基本的なアルゴリズムの紹介からその応用編を終えたところで一応の区切りとさせていただくことになった．名前だけは有名だが，その実体を知る者は少ないというアルゴリズムや，プログラミングをするうえでいずれは必ずお世話になるであろうデータ構造の基本などを，できるだけ平易な説明で，しかも誌上だけの知識にならないように具体的なプログラミング例をあげて紹介することを心がけてきたつもりだが，いかがだったろうか．

最後までお付きあいくださった読者の方々に深くお礼申し上げるとともに，辛抱強く連載の面倒を見ていただいた編集部の鈴木氏に感謝して筆（筆者の場合には親指キーボードだが）を置くことにする．また新しい舞台で機会をいただいて，さらなる駄文をお目にかけることができれば幸いである．というわけで，実践C塾アルゴリズム編は，ひとまずこれにて一件落着．

プログラミングに王道なし．しかし，うまい手はある．



プログラムリスト

先月のBR.EXEに「しおり」の挿入・削除の機能を追加するリストをリスト1～5に示す．先月掲載したリストのうち，変更のなかったCOMMON.H／BROWS.H／SCREEN.Hは，割愛させていただいた．



リスト1　TEXT.H

```
/* ========================================================================
     TEXT.H              Copyright 1992 bY JUSTECH-LABO. (H.kanoshima)
                                (TEXT.C に含まれる関数のプロトタイプ宣言)
========================================================================= */
LINE    *AskNextLineAddress(LINE *);
LINE    *AskPrevLineAddress(LINE *);
LINE    *AskTopLineAddress(void);
LINE    *AskEndLineAddress(void);
char    *AskBodyAddress(LINE *);
void    ReadNextText(void);
BOOL    InitializeText(char *, int);

/* ===== V.2 Start =================== */

LINE    *InsertMarkHere(LINE *);
LINE    *DeleteMarkHere(LINE *);

/* ===== V.2 End ===================== */

```

リスト2 BR.C

```
/* ========================================================================
    実践C塾・テーマツール「BR.EXE」    メインソース

    BR.C            Copyright 1992 bY JUSTECH-LABO. (H.kanoshima)

======================================================================== */
#include    "common.h"
#include    "screen.h"
#include    "text.h"
#include    "brows.h"

#define     ERROR_RETURN        1
#define     NORMAL_RETURN   0

#define     LINETOP             2   /* 画面最上行 */
#define     LINEBOTOM       12      /* 画面最下行 */

#define     SWITCHCHAR          '-'

static      int   TabStep = 4;

/* =========================================================================
 * 起動オプションを処理する。現在処理しているのは't'（ＴＡＢステップの設定）
 * のみである。
 * =========================================================================
*/

static  void    ProcessOption(int argc, char **argv)
{
    argv++;
    while(--argc)
    {
    char    *Option;

        Option = *argv++;
        if(*Option++ == SWITCHCHAR)
        switch(*Option++)
        {
            case    't':
            case    'T':
                    TabStep = atoi(Option);
                    break;
            default:
                    break;
        }
    }
}

/* ==============================
 * 言わずと知れたメインルーチン。
 * ==============================
*/

int     main(int argc, char **argv)
{
    if(argc < 2)
    {
        puts("Usage:br filename.ext [-tn]");
        return  ERROR_RETURN;
    }

    ProcessOption(argc, argv);

    InitializeScreen(LINETOP, LINEBOTOM, TabStep);

    if(InitializeText(argv[1], LINEBOTOM - LINETOP) == BFALSE)
        return  ERROR_RETURN;

    BrowsText(argv[1]);

    return  NORMAL_RETURN;
}
```

リスト 3 BROWS.C

```
/* ========================================================================
    実践C塾・テーマツール「BR.EXE」    ブラウザメイン処理ソース

    BROWS.C         Copyright 1992 bY JUSTECH-LABO. (H.kanoshima)

======================================================================== */
#include    <stdio.h>

#include    "common.h"
#include    "screen.h"
#include    "text.h"
#include    "brows.h"

#define     ESCAPEKEY 0x1b

#define     SCROLLDOWN  ('E' - '@')
#define     SCROLLUP    ('X' - '@')

/* ========= V.2 Start ===================== */

#define     INSERTMARK  ('I' - '@')
#define     DELETEMARK  ('D' - '@')

/* ========= V.2 End ======================= */

/* ============================================================
 * ブラウザのメインルーチン。無入力時の残りテキストの読み込みは、
 * ここで処理している (ReadNextText())
 * ============================================================
*/

void    BrowsText(char *FileName)
{
LINE    *DisplayLineAddress;
BOOL    ReWrite;

    DisplayGuide(FileName);
    DisplayLineAddress = AskTopLineAddress();
    ReWrite = ReWriteFullScreen(DisplayLineAddress);

    for(;;)
    {
        if(kbhit())
        {
        LINE    *lptr = NULL;
        int     Key;

            Key = getch();
            if(Key == ESCAPEKEY)    break;
            switch(Key)
            {
                case    SCROLLDOWN:
                        lptr = AskPrevLineAddress(DisplayLineAddress);
                        break;

                case    SCROLLUP:
                        lptr = AskNextLineAddress(DisplayLineAddress);
                        break;

        /* ========= V.2 Start ===================== */

                case    INSERTMARK:
                        lptr = InsertMarkHere(DisplayLineAddress);
                        break;

                case    DELETEMARK:
                        lptr = DeleteMarkHere(DisplayLineAddress);
                        break;

        /* ========= V.2 End ======================= */

                default:
                        break;
            }
            if(lptr != NULL)
            {
                DisplayLineAddress = lptr;
                ReWrite = ReWriteFullScreen(DisplayLineAddress);
            }
            else
            {
            if(ReWrite == BTRUE)
                    ReWrite = ReWriteFullScreen(DisplayLineAddress);
            }
        }
        ReadNextText();
    }
}
```

ワンラインNEWS
ロータスは自社の表計算ソフト「1-2-3」のワークシートをリアルタイムで稼動させるソフト「リアルタイム日本語版」を開発，出荷を開始した。

リスト4　TEXT.C

```c
/* ========================================================================
    実践C塾・テーマツール「BR.EXE」　　テキスト処理関連ソース

    TEXT.C          Copyright 1992 bY JUSTECH-LABO. (H.kanoshima)

======================================================================== */
#include    <stdio.h>

#include    "common.h"
#include    "screen.h"
#include    "text.h"

#define     LINEMAXLENGTH   80
#define     LINEMAXLENGTHZ          (LINEMAXLENGTH + 1)
#define     LINESIZE                (sizeof(LINE))
#define     PATHLENGTH              80
#define     NEWLINEFLAG             0x00
#define     CONTLINEFLAG            0x01

/* ===== V.2 Start =================== */

#define     MARKCHAR                '¥n'
#define     MARKSTRING              "<---------->"

/* ===== V.2 End ===================== */

LINE    *TopLine, *EndLinePtr, *CurLinePtr;
char    EditFileName[PATHLENGTH];

FILE    *FilePtr;
BOOL    ExistNextText;

/* ======================================
 * ヘッダも含めた一行分のメモリを獲得する。
 * ======================================
 */

static  LINE  *AllocateLine(int BodySize)
{
    return  (LINE *)malloc(LINESIZE + BodySize);
}

/* ===== V.2 Start =================== */

/* ========================
 * 一行分のメモリを解放する。
 * ========================
 */

static  void    FreeLine(LINE *FreeLinePtr)
{
    free(FreeLinePtr);
}

/* ===== V.2 End ===================== */

/* ================================
 * カレント行ポインタをひとつ進める。
 * ================================
 */

static  void    MoveCurLinePtrNext(void)
{
    if(CurLinePtr->NextLinePtr == NULL)   return;
    CurLinePtr = CurLinePtr->NextLinePtr;
}

/* ========================
 * つぎの行のポインタを得る。
 * ========================
 */

LINE    *AskNextLineAddress(LINE *LinePtr)
{
    return  LinePtr->NextLinePtr;
}

/* ======================
 * 前の行のポインタを得る。
 * ======================
 */

LINE    *AskPrevLineAddress(LINE *LinePtr)
{
    return  LinePtr->PrevLinePtr;
}

/* ========================
 * 先頭の行のポインタを得る。
 * ========================
 */

LINE    *AskTopLineAddress(void)
{
    return  TopLine;
}

/* ========================
 * 最後の行のポインタを得る。
 * ========================
 */

LINE    *AskEndLineAddress(void)
{
    return  EndLinePtr;
}

/* ========================================
 * 行ポインタから文字列の先頭アドレスを得る。
 * ========================================
 */

char    *AskBodyAddress(LINE *LinePtr)
{
    if(*LinePtr->Body == MARKCHAR)
        return  MARKSTRING;
    else
        return  LinePtr->Body;
}

/* ==============================
 * 指定された文字列を一行追加する。
 * ==============================
 */

static  BOOL    AddLine(char *StringPtr)
{
LINE    *LinePtr;

    LinePtr = AllocateLine(strlen(StringPtr));
    if(LinePtr == NULL)     return  BFALSE;

    strcpy(LinePtr->Body, StringPtr);

    if(TopLine == NULL)
    {
        TopLine = LinePtr;
        LinePtr->NextLinePtr = NULL;
        LinePtr->PrevLinePtr = NULL;
        CurLinePtr = TopLine;
    }
    else
    {
        LinePtr->NextLinePtr = NULL;
        LinePtr->PrevLinePtr = EndLinePtr;
        EndLinePtr->NextLinePtr = LinePtr;
    }
    EndLinePtr = LinePtr;

    return  BTRUE;
}

/* ==================================================
 * バイト数指定、ただし漢字が１バイト目でちぎれないように
 * ファイルから一行読み込む。ＴＡＢはそれに該当する分だけ
 * 空白を読み込んだことにしてバイト数を計算する。
 * ==================================================
 */

static  char  *jfgets(char *BufferPtr, int Length, FILE *FilePtr)
{
int     TabStep = AskTabStep();
int     ByteCount = 0;
char    *BufferTop = BufferPtr;
BOOL    KanjiFlag = BFALSE;

    Length--;

    while(ByteCount < Length)
    {
    int     c;

        c = fgetc(FilePtr);
        switch(c)
        {
            case    EOF:
                        if(ByteCount == 0)
                            return  NULL;
                        else
                        {
                            *BufferPtr = '¥0';
                            return  BufferTop;
                        }
            case    '¥n':
                        *BufferPtr = '¥0';
                        return  BufferTop;
            case    '¥t':
                        *BufferPtr++ = c;
                        ByteCount += TabStep - (ByteCount % TabStep);
                        KanjiFlag = BFALSE;
                        break;
            default     :
                        if(!iskanji(c) || KanjiFlag == BTRUE)
                        {
                            *BufferPtr++ = c;
                            ByteCount++;
                            KanjiFlag = BFALSE;
                        }
                        else
                        {
                            if(Length - ByteCount < 2)
                            {
                                ungetc(c, FilePtr);
                                *BufferPtr = '¥0';
                                return  BufferTop;
                            }
                            else
                            {
                                *BufferPtr++ = c;
                                ByteCount++;
                                KanjiFlag = BTRUE;
                            }
                        }
        }
    }
    *BufferPtr = '¥0';
    return  BufferTop;
}

/* ========================================
```


ワンラインNEWS
リコーはバーコードでパソコンが操作できるソフト「マイツールバーコード」を発売．キーボードから入力するデータをバーコードに割り当て，読み取ることができる．

```
 * ファイルから一行読み込んでリストに追加する。
 * =============================================
 */

void    ReadNextText(void)
{
char    ReadBuffer[LINEMAXLENGTHZ];

    if(jfgets(ReadBuffer, LINEMAXLENGTHZ, FilePtr) == NULL)
    {
        fclose(FilePtr);
        ExistNextText = BFALSE;
    }
    else
    {
        if(AddLine(ReadBuffer) == BFALSE)
        {
            ExistNextText = BFALSE;
            fclose(FilePtr);
        }
        else
            MoveCurLinePtrNext();
    }
}

/* ===================================================================
 * ファイルから表示範囲に相当する行数だけ読み込んでリストに追加する。
 * ===================================================================
 */

static  void    ReadInitialText(int InitialReadCount)
{
char    ReadBuffer[LINEMAXLENGTHZ];

    ExistNextText = BTRUE;

    while(jfgets(ReadBuffer, LINEMAXLENGTHZ, FilePtr) != NULL)
    {
        if(!InitialReadCount--)
            break;
        if(AddLine(ReadBuffer) == BFALSE)
        {
            ExistNextText = BFALSE;
            fclose(FilePtr);
            break;
        }
        MoveCurLinePtrNext();
    }
}

/* =========================================================================
 * このモジュールを初期化する。主な処理は変数の初期化、ファイルのオープンと
 * 表示範囲分の行数の読み込み。
 * =========================================================================
 */

BOOL    InitializeText(char *FileName, int InitialReadCount)
{
    strcpy(EditFileName, FileName);
    FilePtr = fopen(EditFileName, "rt");    /* テキストモードリード */
    if(FilePtr == NULL)
    {
        printf("ファイルがありません(%s)¥n", FileName);
        return  BFALSE;
    }
    EndLinePtr = CurLinePtr = TopLine = NULL;

    ReadInitialText(InitialReadCount);

    return  BTRUE;
}

/* ===== V.2 Start =================== */

LINE    *InsertMarkHere(LINE *InsertLinePtr)
{
LINE    *NewLinePtr;

    NewLinePtr = AllocateLine(sizeof MARKCHAR);
    if(NewLinePtr == NULL)      return  NULL;

    *NewLinePtr->Body = MARKCHAR;

    if(InsertLinePtr->PrevLinePtr == NULL)
    {
        TopLine = NewLinePtr;
        NewLinePtr->NextLinePtr = InsertLinePtr;
        NewLinePtr->PrevLinePtr = NULL;
        InsertLinePtr->PrevLinePtr = NewLinePtr;
    }
    else
    {
        InsertLinePtr->PrevLinePtr->NextLinePtr = NewLinePtr;
        NewLinePtr->NextLinePtr = InsertLinePtr;
        NewLinePtr->PrevLinePtr = InsertLinePtr->PrevLinePtr;
        InsertLinePtr->PrevLinePtr = NewLinePtr;
    }
    return  NewLinePtr;
}

LINE    *DeleteMarkHere(LINE *DeleteLinePtr)
{
LINE    *NewLinePtr;

    if(*DeleteLinePtr->Body != MARKCHAR)
        return  NULL;

    if(DeleteLinePtr->PrevLinePtr == NULL)
    {
        TopLine = DeleteLinePtr->NextLinePtr;
        DeleteLinePtr->NextLinePtr->PrevLinePtr = NULL;
        NewLinePtr = DeleteLinePtr->NextLinePtr;
    }
    else
    {
        DeleteLinePtr->PrevLinePtr->NextLinePtr = DeleteLinePtr->NextLinePtr;
        DeleteLinePtr->NextLinePtr->PrevLinePtr = DeleteLinePtr->PrevLinePtr;
        NewLinePtr = DeleteLinePtr->NextLinePtr;
    }
    FreeLine(DeleteLinePtr);
    return  NewLinePtr;
}

/* ===== V.2 End ===================== */

```



リスト5　SCREEN.C

```c
/* ======================================================================
    実践C塾・テーマツール「BR.EXE」    表示処理関連ソース

    SCREEN.C              Copyright 1992 bY JUSTECH-LABO. (H.kanoshima)

====================================================================== */
#include    <stdio.h>

#include    "common.h"
#include    "text.h"
#include    "screen.h"

#define     SCREENWIDTH     80

int     ScreenY0, ScreenY1, ScreenHight, ScreenWidth;

int     TabStep = 8;

/* ====================
 * ＴＡＢステップを返す。
 * ====================
 */
int     AskTabStep(void)
{
    return  TabStep;
}

/* ==========================================================
 * エスケープシーケンス出力用。先に0x1bを出力し、つぎに与えられた
 * 文字列を出力する。
 * ==========================================================
 */

static  void    Escape(const char *Sequence)
{
    putchar(0x1b);
    printf(Sequence);
}

/* ============================================
 * 指定行にカーソルを移動する。桁は１に固定される。
 * ============================================
 */
void    CsetY(int y)
{
char    Sequence[16];

    sprintf(Sequence, "[%d;%dH", y, 1);
    Escape(Sequence);
}

/* =====================================================
 * 表示範囲の最上行にカーソルを移動する。桁は１に固定される。
 * =====================================================
 */

void    CsetTopLine(void)
{
    CsetY(ScreenY0);
}

/* =====================================================
 * 表示範囲の最下行にカーソルを移動する。桁は１に固定される。
 * =====================================================
 */

void    CsetBotomLine(void)
{
    CsetY(ScreenY1);
}

/* ==========================================================
 * 表示範囲の最上行－１にカーソルを移動する。桁は１に固定される。
 * 上ガイドライン表示のため。
 * ==========================================================
 */

void    CsetTopGuideLine(void)
{
    CsetY(ScreenY0 - 1);
}

/* ==========================================================
 * 表示範囲の最下行＋１にカーソルを移動する。桁は１に固定される。
 * 下ガイドライン表示のため。
 * ==========================================================
 */

void    CsetBotomGuideLine(void)
{
    CsetY(ScreenY1 + 1);
}

/* ================================
 * 現カーソル位置から右端までを消去する。
 * ================================
 */

void    EreaseToLineEnd(void)
{
    Escape("[0K");
}

/* ========================================================
 * 文字列を表示する。ＴＡＢステップが可変のため、ＴＡＢはそれに
 * 該当するだけの半角空白に置き換える。
 * ========================================================
 */

void    PutString(char *StringPtr)
{
int     TabStep = AskTabStep();
int     Column = 0;

    while(*StringPtr != '¥0')
    {
        if(*StringPtr == '¥t')
        {
        int     i;

            Column += (i = TabStep - (Column % TabStep));
            for(; i; i--)
                putchar(' ');
        }
        else
        {
            putchar(*StringPtr);
            Column++;
        }
        StringPtr++;
    }
}

/* ==========================================================
 * 表示範囲にある全行を表示する。途中でキー入力があった場合は
 * そこで中断してBFALSEを返し、それ以外はBTRUE を返す。
 * ==========================================================
 */

BOOL    ReWriteFullScreen(LINE *LinePtr)
{
int     LineCount;

    CsetTopLine();
    LineCount = ScreenHight;

    while(LineCount--)
    {
    char    *StringPtr;

        if(LinePtr != NULL)
        {
            PutString(AskBodyAddress(LinePtr));
            LinePtr = AskNextLineAddress(LinePtr);
        }
        EreaseToLineEnd();
        putchar('¥n');

        if(kbhit())   return   BTRUE;
    }
    return  BFALSE;
}

/* ===================================
 * 表示範囲上下のガイドラインを表示する。
 * ===================================
 */

void    DisplayGuide(char *FileName)
{
int   Count;

    CsetTopGuideLine();
    printf("[ %s ] ", FileName);
    Count = ScreenWidth - strlen(FileName) - 5;
    while(Count--)
        putchar('-');
    CsetBotomGuideLine();
    Count = ScreenWidth;
    while(Count--)
        putchar('-');
}

/* ============================
 * 本モジュールの初期値を設定する。
 * ============================
 */

void    InitializeScreen(int y0, int y1, int tabstep)
{
    TabStep = tabstep;

    ScreenY0 = y0 + 1;
    ScreenY1 = y1 - 1;

    ScreenWidth = SCREENWIDTH;
    ScreenHight = ScreenY1 - ScreenY0 + 1;
}
```

98活用アセンブラ技法

キーボード編
第2回

キーボードドライバの作成

星野 操

前回はキーボードからのキー入力割り込みをトラップし，それを参照するドライバを作成した．簡単な処理はこのようなドライバでも可能だが，どうしてもリスクが残ってしまう．安全さと自由度を増すために，今回はキーボードドライバ自身を作成する．

かなり突然な話ですが，本誌がプログラマーズ・ページとしては一応終了することになり，それに伴って本連載も今回で終了することになりました．いろいろ書き足りないこともないわけではないのですが，ともかくキーボード周りについて形を整えることにします．

前回はキーボードからのキー入力割り込みをトラップし，それを参照するというドライバを作成しました．簡単な処理はこのようなドライバでも可能ですし，またそのほうがシンプルにもなります．しかし自由度はそれほど上がりませんし，どうしてもリスクが多少残ってしまいます．

より安全に，そして便利なものにしようとすると，キーボードドライバそれ自体を書き直す必要がでてきます．そこで今回は，キーボードドライバを書いてみることにします．まず，キーボード関係のワークエリアなどについて見ていくことにしましょう．

キーボードBIOSの構造

キーボードBIOS（およびそれに伴うワークエリア）の構造は，ノーマルモードとハイレゾモードでかなり異なっています．タイマBIOSと同様に，ハイレゾでBIOSを書き直す時に全面的に手を入れたわけです．それだけ手を入れたくなるBIOSだったというわけなのでしょうが…….

とはいえ，ノーマルモードのBIOSもさほど機能的に低いというわけではありません．IBM PCのキーボードBIOSと比較すると，ほぼ同レベルのものを備えています（ただし，ATのBIOSには多少劣りますが）．

前号で記述したように，キーの状態に変化があると，キーボードからシリアルラインを経由してデータが転送されてきます．転送されるデータは各キーに一意的に付けられたキーコードと，キーのメーク・ブレークを示す最上位１ビットです．ノートなど一部の機種にあるNUMシフトキーを除いて，シフト状態などはすべてソフトウェアでコントロールするようになっています．ですからシフト状態はワークエリアに保存し，キーコードが到着した時点で適切にエンコードすることになります．WSのキーボードには，完全に内部的にキーをエンコードし，アスキーコードを直接送出してくるものがありますが，それよりは自由度が高く，その分煩雑であるということになります．

ノーマルモードBIOSでは，エンコードされたデータは下位メモリのワークエリアに記憶されます．キーデータ自体を記憶する部分はリングバッファ構造になっていて，16ワードの領域が用意されています．ここでキーのスキャンコード１バイトとエンコードされたアスキーコード１バイトをペアで記憶させるため，最高16文字の先打ちが可能となっています．スキャンコードはたいていのキーではキーコードと同一ですが，アスキーコードが０となる[Fx]や[ins][del]などの特殊キーでは，シフト状態によってキーコードとは異なる値が与えられます．これにより，特殊キーでもある種のシフトコンビネーションで使い分けを行うことが可能となります．

入力されたキーデータはバッファの末尾ポインタの示す領域に挿入され，末尾ポインタは１つ後ろにずれます．逆にデータを取り出す時には，先頭ポインタの部分からデータを取り出し，先頭ポインタを後ろに進めます．ポインタ更新時に領域の末端を超えると，先頭に値を設定し直します．このようにすることでバッファをリング状に使用できるのです．98の場合，これらのポインタとは別に，データカウンタを示すバイト変数が用意されています．

IBM PCの場合もキーボードバッファはほぼ同様な構造になっています．ただし，IBM PCはリングバッファを管理する変数として，その先頭ポインタと末尾ポインタしか持っていません．この時取れる偏差は０～15となりますから，16文字を格納してバッファをフルにさせることはできません．15文字の先打ちが限度となります．それに対して98の場合は，別にバッファ内の文字カウンタをワークエリアに用意しているので，16文字満杯に詰めても，それとバッファが空の状態とを区別できるのです．

つまり１文字分余分にバッファを使えるわけですが，問題がないでもありません．万一の事故でポインタの内容がおかしくなった場合，IBMのような方法だとゴミデータがいくつか入力されるかもしれませんが，最終的には正常な状態に落ち着きます．これは，先頭・末尾ポインタが一致している状態をバッファが空の状態とし，その状態になるまでデータを取り出し，ポインタを更新するからです．これに対して98の場合，文字カウンタが０になれば，後はデータがないということになりますからポインタは更新されません．この時に先頭・末尾のポインタがずれていると，以後何文字か遅れて入力したデータが出てくるということになってしまいます．余分な領域を使った分だけ，不安定になってしまうわけです．

ワンラインNEWS
米アップルコンピュータは，小型コンピュータを開発する新組織を発表した．名称は「パーソナル・インタラクティブ・エレクトロニクス」で，略してアップルパイ．

エラーカウンタ／ビットマップテーブル

さて，これらのほかに98の場合は，キーボードのエラーカウンタとビットマップテーブルが用意されています．エラーカウンタはキーボードのデータ受信時にエラーが発生した回数を記憶するもので，ROM-BIOSではこれが一定数に達するとエラーを無視して正常に受け付けられたものとして取り扱います．ただし，現実問題としてデータエラーがそう頻繁に起こることは考えられません．またハンドシェークをきちんと実行しているため，データの取りこぼしをすることもほとんど考えられません．つまり，エラーが数回以上続けて生じた場合には，キーボード関係のハードウェアの設定が異常な状態に陥っていると考えるのが自然でしょう．実際には回数を無視してリトライしても，ほとんど影響はないと思われます（キーボードリセットを掛けない限り，その後正常に動作するとは思えない）．

ビットマップテーブルはIBM PCには用意されていないものですが，現在のキーボードの状況を示すものです．各キーに1ビットが割り当てられていて，対応するキーが押されている場合には，当該ビットが1になっています．ですから，あるキーが押されているかどうかを判断する場合には，キーコードやキーバッファの内容を参照しなくても，この領域を見ているだけで判断できます．ただし，ROM-BIOSではキーのメーク・ブレークビットは参照しておらず，対応するビットを反転しているだけですので，なんらかの事故によってこれが一回反転してしまうと，以後ずっと状態が反転したままになってしまいます．

同様な問題はシフト状態テーブルについてもいえます．シフト状態テーブルは，[SHIFT] [CAPS] [CTRL] [カナ] [GRAPH] の5つのキーの押し下げ状態を格納しているものです．実際にはこれらは，70～74のキーコードを割り当てられていて，75～77に該当するキーは存在していないので，まったく同一の内容がビットマップテーブルの中にもあることになります．ビットマップテーブルと同様にシフト状態が反転すると，それはずっとそのままになってしまいます．これはビットマップテーブルより影響が大で，たとえば [CTRL] キーを押したままにしないと，通常の文字入力ができなくなったりします．これらは，シフトキーなどを押した状態でキーボードドライバなどを付けたり外したりした場合に発生することがあるようです．

同一のキーコードがきても，シフト状態に依存してエンコードした結果を変化させないといけないわけですから，そのための処理が必要になります．ROM-BIOSでは割り込み時の処理を簡単にするために，エンコード用のテーブルのオフセット値を，やはり下位メモリにセットするようになっています．ただし，この領域はワードサイズしか用意されていないので，セグメントを指定することができません．あくまでROM-BIOSの中のテーブルを設定できるだけであり，任意のテーブルアドレスを設定してキーボードの配置を変える（たとえば，ASCII配列に変更する）ことはできません．ある種のキーボードドライバでは，自分の変換テーブルのアドレスをこのワークエリアに設定していましたが，あくまでROM-BIOSのためのエリアであるので，あまり意味のある作業ではありません．

キーボードBIOSの代替処理

キーボードドライバを作成するためには，送られてきたデータによって以上のような各ワークエリアの内容を処理する必要があります．それに加えて，必要に応じてキーボードBIOS自体の代替処理もしなければいけません．前記のように，キーボートBIOSの内容自体はIBM PCでも98でもあまり変わりはありません．

98の場合，キーボードBIOSはINT18で起動されます．ただし，この割り込みはキーボード専用ではなく，システムコントロールや画面処理，グラフィック処理などにも使用されます．キーボードBIOSは，AHレジスタで示されるファンクションコードが最下位の部分にマップされています．

機能0でキーバッファからデータを取り出します．これは，データが発生するまで返ってこないファンクションです．データは，AXレジスタに返されます．実際には，キーバッファの内容を取得するだけです．もちろん，先頭ポインタとデータカウンタをアップデートしなければいけません．

BIOSにいったきりになるのを避けるために，ファンクション1でバッファの状態をチェックすることができます．データがすでに存在している場合には，ファンクション0と同様にAXレジスタにデータが入りますが，このほかにBHレジスタが1に設定されます．データがない場合にはBHが0になり，すぐに復帰します．IBM PCでは同様なファンクションはBHレジスタを使用せずにZフラグを使っています．基本的には，よけいなレジスタを破壊しないIBM PCのアプローチのほうが正当といえるでしょう．

リスト1 KEYBIOS.ASM

```
        .model  small,pascal
        include condx.inc
        masm51
        .286
        .stack  2000h
        .code
        public  my_id,interrupt_list,init_func,release_func
        public  check_func,intf_func
my_id   dw      258eh
int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends
interrupt_list  label   byte
i09     int_work        <09h,xint09,0>
i18     int_work        <18h,xint18,0>
        dw      0
;――――――――――――――――――――――
; hardware constants definition
;――――――――――――――――――――――
KEY_COMMAND     equ     43h
KEY_STATUS      equ     43h
KEY_DATA        equ     41h
KEY_OK          equ     16h
KEY_ERROR       equ     14h
KEY_ERROR_MASK  equ     38h
PIC             equ     0
PIC_MASK        equ     2
EOI_COMMAND     equ     20h
;――――――――――――――――――――――
; keyboard data block
;       for extended service
;――――――――――――――――――――――
default_offset  equ     502h
        .data
        ifndef  MY_BUFFER
my_key_buffer_size      equ     16
key_buffer_address      equ     <system_key_buffer>
        else
my_key_buffer_size      equ     32
my_key_buffer   dw      my_key_buffer_size dup (?)
key_buffer_address      equ     <my_key_buffer>
        endif
key_data_block  label   byte
key_shift_code  db      70h,71h,72h,73h,74h,0ffh,0ffh,0ffh
key_buffer      dd      key_buffer_address
key_entry_table dd      my_key_entry_table
key_int_table   dd      my_key_int_table
key_buffer_size dw      my_key_buffer_size
my_key_int_table        db      2,60h,0e1h
my_key_entry_table      dd      my_trans_table
        db      0,1,2,3,4,5,4,5,6,6,6,6,6,6,6,6
        db      7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
special_key_offset      db      0,20h,30h,50h,60h,70h,80h,0ffh
SD0     equ     0f8h
SD2     equ     0f9h
SD3     equ     0fah
SD5     equ     0fbh
SD6     equ     0fch
SD7     equ     0fdh
SD8     equ     0feh
SNL     equ     0ffh
my_trans_table  label   byte
        ; normal key ascii data
        db      1bh,"1234567890-^¥",8,9
        db      "qwertyuiop@[",0dh,"asd"
        db      "fghjkl;:]zxcvbnm"
        db      ",",".","/",SNL," ",SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0
        db      "-/789*456+123=0,"
        db      ".",SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; shift key ascii data
        db      1bh,"!""#$%&'()0=~|",8,9
        db      "QWERTYUIOP`{",0dh,"ASD"
        db      "FGHJKL+*}ZXCVBNM"
        db      "<",">","?","_"," ",SD7,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD2,SD0
        db      "-/789*456+123=0,"
        db      ".",SD5,SD7,SD7,SD7,SD7,SD7,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; caps key ascii data
        db      1bh,"1234567890-^¥",8,9
        db      "QWERTYUIOP@[",0dh,"ASD"
        db      "FGHJKL;:]ZXCVBNM"
        db      ",",".","/",SNL," ",SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0
        db      "-/789*456+123=0,"
        db      ".",SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; shift+caps key ascii data
        db      1bh,"!""#$%&'()0=~|",8,9
        db      "qwertyuiop`{",0dh,"asd"
        db      "fghjkl+*}zxcvbnm"
        db      "<",">","?","_"," ",SD7,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD2,SD0
        db      "-/789*456+123=0,"
        db      ".",SD5,SD7,SD7,SD7,SD7,SD7,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; kana key ascii data
        db      1bh,"ﾇﾌｱｳｴｵﾔﾕﾖﾜﾎﾍｰ",8,9
        db      "ﾀﾃｲｽｶﾝﾅﾆﾗｾﾞﾟ",0dh,"ﾁﾄｼ"
        db      "ﾊｷｸﾏﾉﾘﾚｹﾑﾂｻｿﾋｺﾐﾓ"
        db      "ﾈ","ﾙ","ﾒ","ﾛ"," ",SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0
        db      "-/789*456+123=0,"
        db      ".",SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; kana+shift key ascii data
        db      1bh,"ﾇﾌｧｩｪｫｬｭｮｦﾎﾍｰ",8,9
        db      "ﾀﾃｨｽｶﾝﾅﾆﾗｾﾞ｢",0dh,"ﾁﾄｼ"
        db      "ﾊｷｸﾏﾉﾘﾚｹ｣ｯｻｿﾋｺﾐﾓ"
        db      "､","｡","･","ﾛ"," ",SD7,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD2,SD0
        db      "-/789*456+123=0,"
        db      ".",SD5,SD7,SD7,SD7,SD7,SD7,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SD5,SNL
        db      SNL,SNL,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SD2,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; graphic key data
        db      1bh,SNL,SNL,SNL,SNL,0f2h,0f3h,0f4h,0f5h,0f6h,0f7h,8ch,8bh,0f1h,8,19h
        db      9ch,9dh,0e4h,0e5h,0eeh,0efh,0f0h,0e8h,0e9h,08dh,08ah,SNL,0dh,9eh,9fh,0e6h
        db      0e7h,0ech,0edh,0eah,0ebh,8eh,89h,94h,SNL,80h,81h,82h,83h,84h,85h,86h
        db      87h,88h,97h,SNL,20h,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SNL,SD0
        db      2dh,2fh,98h,91h,99h,95h,0e1h,0e2h,0e3h,0e0h,93h,8fh,92h,96h,9ah,90h
        db      9bh,SD0,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        ; control key data
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,1eh,1ch,08h,09h
        db      11h,17h,05h,12h,14h,19h,15h,09h,0fh,10h,00h,1bh,0dh,01h,13h,04h
        db      06h,07h,08h,0ah,0bh,0ch,SNL,SNL,1dh,1ah,18h,03h,16h,02h,0eh,0dh
        db      SNL,SNL,SNL,1fh,20h,SD8,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SD0,SNL,SD0
        db      2dh,2fh,37h,38h,39h,2ah,34h,35h,36h,2bh,31h,32h,33h,3dh,30h,3ch
        db      2eh,SD6,SD8,SD8,SD8,SD8,SD8,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
        db      SNL,SNL,SD3,SD3,SD3,SD3,SD3,SD3,SD3,SD3,SD3,SD3,SNL,SNL,SNL,SNL
        db      SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL,SNL
;――――――――――――――――――――――
; low memory structure
;――――――――――――――――――――――
low_memory      segment at      0
        org     500h
system_flag             db      ?
        org     502h
system_key_buffer       dw      16 dup (?)
system_dummy            dw      ?
system_head_pointer     dw      ?
system_tail_pointer     dw      ?
system_key_counter      db      ?
system_retry_counter    db      ?
system_status_table     db      16 dup (?)
system_shift_status     db      ?
low_memory      ends
        .code
;――――――――――――――――――――――
; time waste function
;――――――――――――――――――――――
noop    proc
        push    cx
        mov     cx,50h
        loop    $
        pop     cx
        ret
noop    endp
;――――――――――――――――――――――
; keyboard hardware interrupt routine
;――――――――――――――――――――――
xint09:
        sti
        pusha
        push    ds
        push    es
        mov     ax,_DATA
        mov     ds,ax
        ;――――――――――――――――――――――
        ; check error
        ;――――――――――――――――――――――
        in      al,KEY_STATUS
        and     al,KEY_ERROR_MASK
        @if     nz
                ; some error occurs
                call    noop
                ; retransmit flag set
                mov     al,KEY_ERROR
                out     KEY_COMMAND,al
                call    noop
                ; clear rx-ready
                in      al,KEY_DATA
                ; and return
                jmp     shortcut
        @endif
        ;――――――――――――――――――――――
```


ワンラインNEWS
テクネは，キヤノンのインクジェット方式のカラー複写機をアップルコンピュータの「Macintosh」で制御するソフト「BJ Master Jr」を発売した。

ただMS-Cのライブラリ（あるいはその仕様を引き継いだTCのライブラリ）のINT86を使用する場合には，フラグはキャリーフラグしか参照できません．ですから，Zフラグを使用するよりは，レジスタを使用する98の仕様のほうが便利だといえます．

ファンクション2は前記のシフトワークエリアを取得するもの，4はビットマップテーブルの内容を取得するものです．3はキーボード関係のインターフェースとワークエリアを初期化するもの，5はデータがある時にのみデータを取り出す，ファンクション0と1を合わせたようなものです．

以上のようなファンクションについては，ワークエリアの処理をノーマルモードのBIOSに準じている限り，ROM-BIOSのそれをそのまま使うことができます．しかしたとえばキーバッファを拡張する，というような処理をする場合には，拡張したバッファに対応するBIOSの機能をインプリメントしなおさなければいけません．

ハイレゾモードの拡張

さて，以上でノーマルモードのBIOSについてはほぼ説明が終わりましたが，ハイレゾモードでは，どのような拡張がなされているのでしょうか．まず，およそいろいろなものが再設定可能になっています．ノーマルモードでは［STOP］キーでINT6,［COPY］キーでINT5が発生するようになっていますが，これらを変更することができます．INT5･6を発生させるキーコードを，それぞれ独立に（かつ複数）定義できるようになっているわけです．これらを定義するテーブルのアドレスは下位ワーク（ノーマルモードの500H～のワークエリアではなく，400H～）に設定されます．

また，シフトキーも同様に任意に指定できます．アスキーコードへの変換テーブルも同時に設定できるので，およそ自由なキーコードを発生させることができるようになっているのです．以上のような処理が，ROM-BIOS自身には手を加えることなく，単にテーブルをすり替えるだけで可能なのです．また，キーバッファおよびその大きさも自由に定義できるので，必要に応じて大きなサイズの（タイプアヘッドの効く）バッファを設定することもできます．

もう1つの違いは，キーバッファに格納するデータの形式です．ノーマルモードは前記のようにエンコードした後のスキャンコードとアスキーコードのペアをバッファに格納しました．しかし，ハイレゾBIOSでは，キーコードとシフト状態のペアを格納します．そして，ファンクション0/1などのデータを取り出す機能の中で，エンコード処理を行うのです．もちろん，直接キーコード・シフトコードのペアを（追加されたファンクションで）取得することもできますし，これらのペアからエンコードした結果を別に得ることもできます．

このようにすることで，まずハードウェア割り込み時の負荷を減らすことができます．ただでさえ任意のシフトキー・割り込みキーをサポートするために割り込みの負荷が重くなっているので，割り込み時にエンコード処理をしないというのは有効だと思われます．BIOSを呼び出す時には，ユーザープログラムが動いているわけですから，その処理は多少重くなっても影響は少ないわけです．さらに，常にシフトコードが把握できるという利点もあります．通常キーはもちろん，特殊キーでもかなりのものが複数のシフト状態で同一のキーデータを生成します．ですから，たとえば右カーソルとシフト＋右カーソルをキーデータから区別することはできません．

アプリケーションでこれらを区別する必要がある場合には，BIOSでデータを取得した時点で別にシフト状態を得ることになるわけですが，常時キースキャンをするわけではないので，実際にキーを押した時点とシフト状態が変わっている可能性があります．ハイレゾBIOSでは，このような問題は生じません．

このほかに，ハイレゾモードでは用意されていませんがAT-BIOSにある機能に，キーコードの変換機能があります．AT-BIOSではキーのメーク・ブレークが発生すると，そのキーコードをレジスタに設定し，システム割り込みを起動します．この割り込みをトラップすれば，特定のキーコードを別のキーコードにすり替えたり，無効化したりすることが可能です．たとえば，IBM PCの［CTRL］キーと［CAPS］キーを入れ替えるのは，この機能を使用すれば常駐プログラム1つで簡単に実現できます．

キーボードドライバプログラム

今回のキーボードドライバは，以上のようなハイレゾモードの拡張を取り入れるようにします．ただし，互換性などの面からハイレゾの仕様を完全に受け入れることはできません．すべてのアプリケーションがBIOS経由でデータをアクセスしていればよいのですが，実際には少なからぬソフトがバッファやワークエリアに直接アクセスしています．ですから，たとえばバッフ

```
	; set data ok status
	;-----------------------------
	mov	al,KEY_OK
	out	KEY_COMMAND,al
	call	noop

	;-----------------------------
	; get data, and convert it
	;-----------------------------
	in	al,KEY_DATA
	mov	bl,0ffh
	mov	ah,9
	int	18h

	;-----------------------------
	; check flag
	;-----------------------------
	cmp	al,0ffh
	@if	z
		jmp	shortcut
	@endif

	;-----------------------------
	; first, set key status table
	;-----------------------------
	mov	bx,low_memory
	mov	es,bx
	assume	es:low_memory

	mov	bh,0			; get map offset
	mov	bl,al
	and	bl,7fh
	shr	bx,1
	shr	bx,1
	shr	bx,1

	mov	cl,al			; get bit code
	and	cl,7
	mov	ah,1
	shl	ah,cl

	test	al,80h			; check make/break
	@if	z
		; make, set bit
		or	system_status_table[bx],ah
	@else
		; break, reset bit
		not	ah
		and	system_status_table[bx],ah
	@endif
	assume	es:nothing

	;-----------------------------
	; check interrupt key
	;-----------------------------
	test	al,80h
	@if	z
		; make code, advance
		les	bx,key_int_table

		mov	cl,es:[bx]	; get int table entry count
		@begin
			inc	bx
			mov	ah,al	; check code
			xor	ah,es:[bx]
			and	ah,7fh
			@if	z
				; match, cause interrupt
				mov	al,EOI_COMMAND	; send EOI
				out	PIC,al

				test	byte ptr es:[bx],80h
				pop	es
				pop	ds
				popa

				@if	nz
					int	5
				@else
					int	6
				@endif
				iret
			@endif
			dec	cl
		@until	z
	@endif

	;-----------------------------
	; check shift key
	;-----------------------------
	mov	bx,seg key_shift_code
	mov	es,bx			; get shift table address
	mov	di,offset key_shift_code
	mov	cx,8
	mov	ah,al			; make key code
	and	al,7fh
	repnz	scasb			; and search
	mov	al,ah
	@if	z
		; it's a shift key
		sub	di,offset key_shift_code
		dec	di
		mov	cx,di
		mov	ah,1		; make shift bit
		shl	ah,cl

		mov	bx,low_memory
		mov	es,bx
		assume	es:low_memory
		test	al,80h
		@if	z
			; make code, set bit
			or	[system_shift_status],ah
		@else
			not	ah
			and	[system_shift_status],ah
		@endif
		assume	es:nothing

		; short cut
		jmp	shortcut
	@endif

	;-----------------------------
	; normal key, convert it
	;-----------------------------
	test	al,80h
	@if	nz
		; break, nothing to do more
		jmp	shortcut
	@endif

	mov	bx,low_memory
	mov	es,bx
	assume	es:low_memory
	mov	bl,[system_shift_status]
	mov	ah,9
	int	18h

	cmp	ax,0ffffh
	@if	nz
		; check buffer space
		mov	bx,key_buffer_size
		cmp	[system_key_counter],bl
		@if	ae
			; check low memory flag
			test	system_flag,20h
			@if	z
				; no need to beep
				jmp	shortcut
			@endif

			; buffer full, beep
			mov	ah,17h
			int	18h

			; disable key interrupt, and accept interrupt
			in	al,PIC_MASK
			or	al,1
			out	PIC_MASK,al
			mov	al,EOI_COMMAND
			out	PIC,al

			; wait little
			mov	ah,3
			@begin
				; wait vsync edge
				@begin
					in	al,0a0h
					and	al,20h
				@until	z
				@begin
					in	al,0a0h
					and	al,20h
				@until	nz

				dec	ah
			@until	z

			; stop beep, and enable key interrupt
			mov	ah,18h
			int	18h
			cli
			in	al,PIC_MASK
			and	al,not 1
			out	PIC_MASK,al

			; shortcut return
			jmp	shortcut2
		@endif


		; store into buffer
		cli
		inc	[system_key_counter]
		mov	si,[system_tail_pointer]
		sub	si,default_offset

		assume	es:nothing
		les	bx,key_buffer
		mov	es:[bx+si],ax

		shr	si,1
		inc	si
		cmp	si,key_buffer_size
		@if	ae
			mov	si,0
		@endif
		shl	si,1
		add	si,default_offset
		mov	bx,low_memory
		mov	es,bx
		assume	es:low_memory
		mov	[system_tail_pointer],si
	@endif
	assume	es:nothing

shortcut:
	mov	al,EOI_COMMAND
	out	PIC,al
shortcut2:
	pop	es
	pop	ds
	popa
	iret
```

ァに格納するデータ形式をハイレゾモードのようにキーコード＋シフト状態という形式にすることは困難です．

そこで，次のような方針にすることにしました．

- ハイレゾモードで拡張されたワークエリアは，内部的な領域に保持する．
- これらの構造自体はハイレゾモードに準ずる．
- キーバッファは基本的に下位ワークのそれを利用し，格納するデータはノーマルモード形式とする．
- ただし，キーバッファはハイレゾモードと同様再指定可能とし，これに対応するためにKEY-BIOS自体も置き換える．
- AT-BIOSと同様，キーコード受信後キーコード変換用の割り込みを起動する．また，ハイレゾと同様キーコード＋シフト状態からのデータエンコードも割り込みを経由しトラップ可能にする．

では，以上のような方針で作ったキーボードドライバのリストにそって解説していきましょう．

今回のドライバは，デバイスドライバ形式ではなくres3と合わせて常駐プログラム形式とします．ドライバを補助するタイプのプログラムも，やはり常駐プログラムとして作成します．最初の部分の記述は，res3とのインターフェースのために使用されるものです．ここでは，ハードウェア割り込み09とキーボードBIOS割り込みをトラップするので，その記述があります．

定数領域

定数領域は，主としてI/Oポートのアドレスなどを記述してあります．KEY_OK／KEY_ERRORはキーボードとのハンドシェーク時に信号線レベルを設定するためのシリアルチップへのコマンドです．

default_offsetは，システムキーボードバッファのアドレスオフセットです．今回は冗長性を増すために，リングバッファのアクセスに次のような方法を使っています．まず，リングバッファの先頭・末尾各ポインタのオフセットからこのdefault_offsetの値を減算し，その結果をキーボードバッファの先頭のオフセットに加算してアクセスするのです．

システムキーボードバッファは0:502Hからありますから，上記の処理は502Hを引いて502Hをたすだけなので，結局ポインタの値をそのままオフセット値として使用するのと変わりません．ただ，ユーザーサイドでキーボードバッファを再配置する場合，そのバッファのセグメント内のオフセットを直接これらのポインタワークエリアに設定してしまうと，不都合が生じる(危険性がより高い）ためです．

たとえば，バッファの内容をクリアするコマンドは，ハイレゾBIOSでは追加されていますが，ノーマルモードのBIOSではサポートされていません．ですから，[STOP]キーが押された時に，conデバイスは直接このポインタを書き換えることによって，キーバッファの内容をクリアするものと思われます．この時バッファの先頭アドレスと考えられる定数502Hを直接書き込まれてしまうと，本来の位置とは異なる部分をデータバッファにしてしまうことになります．そこで，前記の処理を行い，キーボードバッファが502Hであると決め打ちをしてワークエリアを触るプログラムが同居したとしても障害の程度は低くなるようにするわけです．

もちろんバッファの再配置をする場合には，直接バッファの中身を覗くプログラムは，オフセットの計算を前記の手続きにしたがって行わなければいけませんし，バッファの存在するセグメントを0と仮定してしまっているプログラムは正しいバッファの内容を取得できません．IO.SYSの中のconデバイスは，この直接アクセスを行っているので，バッファの再配置をすると問題が生じる危険はあります．十分に注意を払う必要があるでしょう．

今回は，シンボルMY_BUFFERが定義されて「いない」と，デフォルトのキーバッファにシステムリザーブのものを使用し，これが定義されている場合にはローカルに用意したものを使用するようになっています．ローカルに再定義する場合には，デフォルトで32キャラクタ分のバッファ領域を用意してありますが，もちろんこれを拡張することは可能です．

ワークエリアブロック

その次に，前述したようにハイレゾBIOSに準拠したワークエリアブロックがあります．このブロックのアドレスは，キーボードBIOSの拡張ファンクションで取得することができます．

まず，シフトキーの定義テーブルがあります．シフト状態バイトの各ビットに対応するキーコードをここで定義します．使用しないビットに対応する部分には0FFHを設定します．ここでは，もちろんデフォルトのシフトキーに対応した70H～74Hのキーコードを設定してあります．あまりシフトキーを変更するということは考えられませんが(ハイレゾBIOSは，本当に冗長に作ってあります)，たとえば[CAPS]キー

```


;============================
; BIOS support functions
;============================

        assume  es:low_memory

;----------------------------
; key data get routine
; input
;       none
; output
;       ax:key code
;----------------------------
bios_get_char   proc
        push    bx

        ; wait till data ready
        @begin
                call    bios_get_status
                or      bh,bh
        @until  nz

        ; pointer and count update
        cli
        dec     [system_key_counter]

        mov     bx,[system_head_pointer]
        sub     bx,default_offset-2
        shr     bx,1
        cmp     bx,key_buffer_size
        @if     ae
                ; round up
                mov     bx,0
        @endif
        shl     bx,1
        add     bx,default_offset
        mov     [system_head_pointer],bx
        sti

        ; and return
        pop     bx
        ret
bios_get_char   endp


;----------------------------
; key data check routine
; input
;       none
; output
;       ax:key code
;       bh:key status
;----------------------------
bios_get_status proc
        ; first check buffer length
        cmp     [system_key_counter],0
        @if     z
                ; no key data, reset pointer
                cli
                mov     ax,system_tail_pointer
                mov     system_head_pointer,ax
                sti

                ; and return
                mov     bh,0
                ret
        @endif

        ; scan key data
        mov     si,[system_head_pointer]
        sub     si,default_offset
        assume  es:nothing
        les     ax,key_buffer
        add     si,ax
        mov     ax,es:[si]

        ; and return
        mov     bh,1
        ret
bios_get_status endp

        assume  es:low_memory

;----------------------------
; key shift get routine
; input
;       none
; output
;       al:shift code
;----------------------------
bios_get_shift  proc
        mov     al,[system_shift_status]
        ret
bios_get_shift  endp


;----------------------------
; key buffer initialize routine
; input
;       none
; output
;       none
;----------------------------
bios_init       proc
        cli
        ; initialize keyboard
        mov     al,3ah
        out     PIC,al
        call    noop
        mov     al,32h
        out     PIC,al
        call    noop
        mov     al,16h
        out     PIC,al

        ; clear workarea
        mov     [system_head_pointer],default_offset
        mov     [system_tail_pointer],default_offset
        mov     [system_key_counter],0
        mov     [system_retry_counter],0
        mov     [system_shift_status],0

        mov     si,0
        @begin
                mov     system_status_table[si],0
                inc     si
                cmp     si,16
        @until  ae

        sti
        ret
bios_init       endp


;----------------------------
; key sense routine
; input
;       al:gourp number
; output
;       ah:bit map status
;----------------------------
bios_key_map    proc
        ; get map table contents
        mov     ah,0
        mov     si,ax
        mov     ah,system_status_table[si]
        ret
bios_key_map    endp


;----------------------------
; key sense and get routine
; input
;       none
; output
;       ax:key code
;       bh:status
;----------------------------
bios_sense_and_get      proc
        ; check buffer
        call    bios_get_status
        or      bh,bh
        @if     nz
                ; data exist
                call    bios_get_char
        @endif

        ret
bios_sense_and_get      endp


;----------------------------
; buffer initialize
; input
;       none
; output
;       none
;----------------------------
bios_buffer_init        proc
        ; clear only buffer pointer only
        cli
        mov     [system_head_pointer],default_offset
        mov     [system_tail_pointer],default_offset
        mov     [system_key_counter],0
        sti
        ret
bios_buffer_init        endp


;----------------------------
; non supported function
; input
;       none
; output
;       none
;----------------------------
bios_invalid_function   proc
        ; none
        ret
bios_invalid_function   endp


;----------------------------
; special functions
; input
;       al:0ffh / get parameter block address
;       al:other / hardware interrupt scan code
; output
;       al=0ffh, ax:bx for key data block address
;       al!=0ffh al:new scan code, 0ffh for null key
;----------------------------
bios_special_function   proc
        cmp     al,0ffh
        @if     z
                mov     ax,_DATA
                mov     bx,offset key_data_block
        @else
                ; nothing
        @endif
        ret
bios_special_function   endp


;----------------------------
```

と［カナ］キーを入れ替える，というような操作も可能ですし，［XFER］キーをカナキーの代わりに使うというようなことも不可能ではありません．［カナ］キーを無効化するだけなら，対応する位置のキーコードを0FFHにしてしまえば済みます．ただ，CAPS/カナなどの内部的にオルタネート処理になっているキーと［CTRL］などの単なるシフトキーとを入れ替えるのは好ましい結果が得られないでしょう．

次に，キーボードバッファのアドレスが定義してあります．DD疑似命令の引数にラベルを取った場合，そのセグメントとオフセットが期待通りに設定されます．key_buffer_addressはデフォルトでsystem_key_bufferに文字マクロ定義されていますが，system_key_bufferの存在するセグメントlow_memory（これは物理アドレスに対応している）がDDによって設定されます．キーバッファを再配置する場合には，まずこのワークエリアの値を変えることになります．なお前述のようなオフセット処理を行うので，設定するアドレスのオフセット値についての制限はとくに存在しません．

key_entry_tableは，キーコードのエンコードに使用する構造体のアドレスを示します．デフォルトではmy_key_entry_tableのアドレスが設定されていますが，この構造体は，最初に変換テーブルのアドレス（ロングワードサイズ），次にシフト状態テーブルの2つの要素によって構成されています．

・シフトキーの定義

シフトキーは，デフォルトで5種，最高で8種定義するとが可能です（ただし，内部的に使用しているので，シフト状態が0FFHになることは許されません）．ですからシフト状態はたとえば32通り存在するわけです．しかしそれぞれの状態によって，まったく異なるアスキーコードをエンコードするというわけではありません．［CTRL］キーが押されている場合，［CAPS］キーの状態に依存せず，エンコードされる結果は一定です．ですから，すべてのシフト状態に対応したエンコードテーブルを作成することは効率的ではありません．

そこで，まずシフト状態から特定のシフトコードを作成し，それを元にエンコードをすることにします．たとえば，［CTRL］キーが押されていれば，ほかのシフトキーの状態に依存せずエンコード結果は同一になりますから，シフト状態バイト10H～1FHはシフトコード7に対応するエンコードテーブルを使用する，というようになっているわけです．エンコードテーブルは，1つ1つが128バイトのエントリを持っており，それぞれがキーコードに対応するアスキーコードになっています．ですから，このシフトコード×128＋キーコードという計算式によって，エンコードテーブルのアスキーコードの内容を得ることができます．

ただし，それが無効なキーコンビネーションや特殊キーである場合もあります．通常のキーの場合スキャンコードはキーコードに一致しますが，特殊キーの場合対応するアスキーコードは常に0で，シフト状態によってスキャンコードが変化します．そこで，ここでは直接キー入力のできない0F8H～0FFHの部分をこれらを表現するために使用します．テーブルの対応する値が0FFHであれば，それは無効キーコンビネーションを示すとします．0F8H～0FEHは特殊キーの場合に使用し，それぞれが（スキャンコード－キーコード）の値を示すようにしてあります．この値は，special_key_offset配列によって示されます．ですからこの差分が10Hである場合，および40Hである場合は表現できないことになります．

・バックスラッシュキャラクタ

最近といってもかなり前からですが，文字コード0FCHで半角のバックスラッシュキャラクタがサポートされており，これもグラフキーとのコンビネーションで入力可能になっています．今回のような方法だと，アスキーコード0FCHは入力できなくなるので，それが必要な場合はエンコードプログラムに手を入れるか，あるいはコード7FHを代わりに使用する，などの手段を講じる必要があります．

・key_int_table

key_int_tableは割り込みを発生させるキーを定義するもので，やはりDwordのアドレスを定義してあります．そこで示される内容は，最初の1バイトがテーブルの大きさ（バイト）で，その後が割り込みを発生させるキーコードの列挙です．ただし，最上位ビットがクリアされているキーコードの場合にはINT6を発生させ，そうでない場合にはINT5を発生させます．デフォルトでは［STOP］キーでINT6，［COPY］キーでINT5を発生させるので，プログラムのような定義になっています．

・key_buffer_size

この領域最後のkey_buffer_sizeはリングバッファの大きさ（キーデータのエントリ数）を示します．デフォルトでは16となります．キーバッファを再配置した場合には，key_bufferとともにこのワークエリアも設定し直す必要があります．

・my_trans_table

my_trans_tableは，あるいは誤りがあるかもしれませんがアスキーコー

```
; key data encode routine
; input
;       al:key code
;       bl:shift status
;         :0ffh special function
; output
;       ah:scan code
;       al:ascii code
;─────────────────────
bios_key_encode proc
        ; check special function
        cmp     bl,0ffh
        @if     z
                call    bios_special_function
                ret
        @endif

        ; save registers
        push    bx

        ; get entry table address
        assume  es:nothing
        les     si,key_entry_table

        ; get shift code, and calc offset
        mov     bh,0
        mov     bh,es:[si+bx+4]
        mov     bl,0
        shr     bx,1
        and     al,7fh
        or      bl,al

        ; get ascii table base
        les     si,es:[si]

        ; get ascii code, and check speciality
        mov     ah,es:[si+bx]
        xchg    ah,al
        cmp     al,SNL
        @if     z
                ; null key code
                mov     ax,0ffffh
        @else
        cmp     al,SD0
        @e_if   ae
                ; get special key correct value
                sub     al,SD0
                mov     bx,offset special_key_offset
                xlat
                add     ah,al
                mov     al,0
        @endif

        ; and return
        pop     bx
        ret
bios_key_encode endp

        assume  es:low_memory


;─────────────────────
; int 18 function table
;─────────────────────
        .data
int18_function  label   word
        dw      bios_get_char
        dw      bios_get_status
        dw      bios_get_shift
        dw      bios_init
        dw      bios_key_map
        dw      bios_sense_and_get
        dw      bios_buffer_init
        dw      bios_invalid_function
        dw      bios_invalid_function
        dw      bios_key_encode
int18_function_count    equ     ($-int18_function)/2
        .code

;─────────────────────
; int 18 support routine
;─────────────────────
xint18:
        ; check range
        cmp     ah,int18_function_count
        @if     ae
                ; graph etc. function
                jmp     i18.int_old
        @endif
        sti

        ; save registers
        push    si
        push    ds
        push    es
        mov     si,_DATA
        mov     ds,si
        mov     si,low_memory
        mov     es,si

        ; and branch
        push    ax
        mov     al,ah
        mov     ah,0
        add     ax,ax
        xchg    ax,si
        pop     ax
        call    int18_function[si]

        ; resume registers and return
        pop     es
        pop     ds
        pop     si
        iret

release_func    proc
        ret
release_func    endp

intf_func       proc
        ret
intf_func       endp

init_seg        segment 'stack'
        assume  cs:init_seg

check_func      proc    far
        xor     ax,ax
        ret
check_func      endp

init_func       proc    far
        ret
init_func       endp

init_seg        ends

        end

```

ドの列挙になっています．入力する場合は，通常キーは指定シフトコンビネーションにして直接キーボードを叩いてしまうのが一番簡単です．前記のようにシフト状態は8つの状態にエンコードされるので，128バイトのテーブルエントリは8つ用意されています．もし，ほかのコンビネーション，たとえば［shift］＋［graph］で特定のキーを発生させたい，というような場合には，新たなシフト状態のシフトコード値と，それに対応する変換テーブルのエントリを追加することになります．もちろん，その場合はテーブル全体を別の領域に再配置する必要が出てくるでしょう．

BIOSシステムワーク

その次には，BIOSのシステムワーク領域が定義されています．ここでは，連載で始めて絶対アドレスに対応したセグメント指定が行われています．絶対アドレスを決定する場合には，segment疑似命令の後にat xxxxとセグメントアドレスを直接指定します．後は，この指定されたセグメントをセグメントレジスタにassumeさせることで，そのセグメント内の変数に対するアクセスに，自動的にプレフィクスを付加させることが可能となります．定義してある内容は前記した通りですが，ROM-BIOSが内部的に使用するテーブルアドレスのオフセットを示すワークはダミー領域としてあります．なおこの領域は，ハイレゾBIOSではほかの目的に使用されています．先頭のsystem_flagというエリアは，キーバッファのオーバーフロー時にビープ音を鳴らすかどうかを示すフラグをサポートするために定義してあります．

・noop

その後にプログラムのコード部分があります．最初の関数noopは，時間つぶしの関数です．キーボードのハンドシェークなどの時にかなり長い時間つぶしが必要になるので，ここでは関数で時間つぶしをしています．この関数は，場合によってはCPUやそのほかの条件によってチューニングが必要になるでしょう．あまり遅いと，ハードウェア割り込み時の応答が遅くなることになります．

ハードウェア割り込み処理ルーチン

次がハードウェア割り込みの処理ルーチンです．最初にレジスタをセーブし，DSにデータセグメントの値を設定します．ここでPUSHA命令を使用していますが，これはデータレジスタしかセーブせず，セグメントレジスタは別にセーブする必要があることに注意してください．個別のレジスタのPUSH命令を書いたほうが安全で簡潔かもしれません．それから前号と同じように，まずステータスのチェックをします．エラーがあれば再送要求を出し，ハンドシェークを終了します．前に述べたように，このドライバでは，エラー回数によるキーボードのリセット処理は実施していません．制御構造を破壊していますが，レジスタの復元を複数の箇所で行うのは不整合の元となるので，ここではJMP命令を使用して出口をできるだけまとめてしまっています．

正常に受信できていれば，データを引き取りキーコードの変換割り込みを起動します．変換割り込みは，キーコードの生成拡張BIOS（AH＝9）をさらに拡張し，BL(シフト状態)が0FFHである場合とします．必要に応じて，この処理ルーチンの中でALに格納されたキーコードの値をすり替えます．AL＝0FFHの時には，そのキー入力は「なかったもの」とされ，すぐに割り込みは終了します．

それから(変換されたかもしれない)キーコードに基づき，ビットマップテーブルの状態を更新します．キーコードをテーブルのオフセットとビット位置とに分解し，そしてMAKE/BREAKフラグに依存してビットをセット／リセットします．この手続きの中で，ESセグメントレジスタにlow_memoryセグメントをassumeすることで，ビットマップテーブルへのセグメントプレフィクスを省略していることに注意してください．もしES(またはDS）に対してlow_memoryセグメントがassumeされていないと，ここで不正なセグメントに対するアクセスであるとエラーが出ます．ですから，命令に直接プレフィクスを記述するより，より安全性の高い記述であるということもできますが，その分煩雑であることは否めません．とくにセグメントレジスタに別の値が設定され無効化された場合に，忘れずにnothingをassumeしないとおかしなことになります．

次に，メークキーであった場合に，割り込み発生のチェックをします．key_int_tableの内容を最上位ビットを無視した形でスキャンし，一致するものがあればテーブル内容の最上位ビットに依存してINT6またはINT5を発生させます．

割り込みキーでなければ，今度はシフトキーのチェックをします．メーク・ブレークフラグをマスクした後rep scasb命令で一気に検索します．ここでデータが見つかった場合，DIレジスタは当該データの次のアドレスを指していることに注意しましょう．ここでは，DIレジスタの値からどのシフ

トビットに対応するシフトキーかを判断し，当該ビットをセットまたはリセットします．シフトキーであれば，やはり以後の処理はショートカットします．よって，リピートをする通常のキーであっても，シフトキーとして取り扱うことが可能です．

なお，［カナ］や［CAPS］などのオルタネートキーがソフト設定できるタイプのキーボードが用意されている機種では，シフト状態を不揮発メモリに書き込むようになっています．これによってマルチタスクシステム使用時やシステム機動時に，シフト状態を復元できるようにしてあります．しかし今回のドライバは現実問題としてマルチタスクシステムで使用することはあまり考えられませんし，シフトキーの再配置があった場合に不都合が生じることになりかねません．そこで今回は，あえて不揮発メモリへのデータの設定をオミットしています．

以上のキーに該当しなくて，かつメークキーであった場合には，エンコードし結果をキーバッファに書き込む必要があります．まず，BLレジスタにシフト状態を設定した後，エンコードのBIOSコールを発行します．そしてその結果が0FFFFHであれば，無効キーコンビネーションだとしてスキップします．そうでなければ，まずバッファの空きエリアをチェックし，バッファフルだとシステム領域のビープフラグをチェックします．これがセットされていれば，ビープ音を出さないといけません．

この間の割り込みを受け付けるために，まずビープセットした後キーボードの割り込みマスクを立ててからEOIコマンドをPICに送出します．これによって以後他のハードウェア割り込みは受け付けるようになります．それから，CRTCのVSYNCビットをチェックし，それによって待ち時間を調節します．これは，決まった時間のディレイを比較的容易に得る方法だと思います．それから，音を止め，割り込みマスクの値を元に戻し，そしてレジスタの内容を復元して復帰します．

バッファに空きがあれば，終端ポインタの指すところにデータを書き込み，それから終端ポインタとデータカウントの更新をします．最後に，EOIをPICに送出して終了します．

KEY-BIOSサポートルーチン

その後は，KEY-BIOSのサポートルーチンです．BIOSの共通エントリでESレジスタの値を0に設定するので，ここでは全般にESはlow_memoryにassumeしてあります．

まずファンクション0のget_charは，get_statusを呼び出し，BH=1になるまで待ちます．それからワークエリアの内容を更新して，終了します．ここでBXレジスタは壊されてはいけないので，保存しておく必要があります．

get_statusはデータカウンタの内容をチェックし，これが0であればBH=0にして，0でなければポインタで示される位置のデータをAXに取得した後BH=1として復帰します．データがない場合，先頭ポインタの値を終端ポインタと同一にします．前述した万一のワークエリアの破壊に対応できるようにするためです．

get_shift／get_key_mapは，それぞれシステムワークエリアの対応するメモリを読み出してレジスタにセットするだけです．bios_initは，キーボードにリセット信号を送り，ワークエリアを初期化します．ただし，シリアルチップの初期化は実行していないので，こちらにトラブルが発生した場合には対処できません．bios_sense_and_getは，バッファ中にデータがあった場合にのみファンクション0相当の処理をし，そうでない場合は1相当の処理をするという便利な機能です．これは，get_statusとget_charを呼び出すことによって実現しています．

拡張ファンクション

ここまでがノーマルBIOSで用意されているファンクションで，以下は拡張ファンクションです．まずbuffer_initはキーバッファのみクリアするファンクションです．ハイレゾBIOSで導入されたもので，このモードではキーバッファの位置がリロケータブルであることから，直接バッファポインタに値を書き込むことができないため，導入されたものと想像されます．今回はオフセットに関しては502Hをそのまま使用することができるので，バッファのアドレスに関係なく，定数を直接書き込んでいます．

invalid_functionは，やはりハイレゾモードでキーデータをエンコードせず，直接キーコードとシフト状態を取り扱うファンクションのためのもので，ここではなにもせずに返っています．ファンクション9は，キーのエンコードファンクションですが，BLが0FFHである場合にのみ，特殊機能と解釈します．この時AL=0FFHであれば，内部のワークエリアのアドレスをAX:BXに設定して復帰します．それ以外の場合は，ハードウェア割り込みの最初のキーコードコンバートのために使用されます．ここでは，なにも設定していませんが，必要に応じてこのファンクションをトラップすることで目的を達成することができます．

BLがそれ以外の値であれば，通常のエンコード処理を行います．まず，key_entry_tableから，シフトコードを算出し，それに128を掛けてtrans_tableのオフセットを計算します．その後キーコードを加算し，アスキーコードを取得します．この値が0F8H～0FFHである場合には，前に書いたように特殊キーとしての処理などを行います．

最後に，int18のキーBIOS処理テーブルと，そのエントリがあります．AH<=9であれば内部処理をし，そうでなければ元の割り込みアドレスにチェインします．

release_func以下は，やはりres3とのインターフェースに使用するもので，基本的に内容は空です．

プログラムの作成

常駐プログラムとして完成させるには，

```
tasm   keybios
tlink  keybios+res3
```

などとすると，keybios. exeができます．単に実行させれば常駐，-rオプションで開放というのはres3を使った常駐プログラムの共通仕様です．

キーボードの拡張方法

キーボード関係になんらかの拡張をしたい，という場合には，このKEYBIOS自体を変更することもできますし，それとは別に常駐プログラムやテーブルの内容を操作することによってKEYBIOS自身はそのままで目的を果たすことも可能です．

たとえば，ローマ字入力をする人々に取っては，[カナ]キーは無用の長物なのだそうです．これを無効化しようという場合，いくつものアプローチが考えられます．まず，KEYBIOSのINT09処理ルーチンに手を加え，[カナ] キーが押された場合にショートカットする方法が考えられます．しかし，目的ごとにKEYBIOSを書き換える必要があり，あまりスマートとはいえません．

これに対して，INT18Hをトラップし，BL!=0FFH，AH=9である場合にAL=72Hか0F2Hであれば，AL=0FFHに設定し直す常駐プログラムを介在させるということも考えられます．つまり，[カナ]キーが押された場合に，そのキーコードを受け取らなかったがごとくに振舞わせるわけです．このようにすれば，無視するキーが複数に増えたとしても，複数の常駐を噛ませるだけでよくなります．また，常駐を終了させれば自動的に元の状態に戻ります．

シフトキーに限っていえば，もっと簡単な方法があります．まず，内部ワークエリアのkey_shift_codeを書き換える方法があります．72Hとあるところを0FFHにしてしまえば，カナキーが入力されてもなにも起こらなくなります．内部ワークのアドレスは，拡張BIOSコールで取得できます．INT18Hのトラップによる無用なオーバーヘッドを避けるためには，このほうが有利です．ただし，元の状態に戻すためには別のプログラムが必要になります．この場合も，常駐化し，init_funcで設定，release_funcで復元というようにしておくと，常駐開放によって自動的に元の状態に戻せます．

さらに，key_entry_tableのシフトコードへの変換テーブルを書き換えることによっても同様な目的を果たせます．常駐プログラムを使う場合には，my_key_entry_tableの中身を書き換えず，自前のkey_entry_tableを作成して，そのアドレスを内部ワークに設定する，という方法もあります．新たに特殊なシフトキーを追加したいというような場合にはこの方法を取るのもよいでしょう．

キー配列をアスキー配列などに変更したいという場合にも，trans_tableの内容を書き換えることで対応できます．永続的な書き換えを期待しない場合には，やはり常駐化しtrans_tableの本体だけを用意しmy_key_entry_tableのベクタを書き換えるという手段を取るのがよいでしょう．もちろん，entry_table全体を用意してもたいして手間は変わりません．

特定のキーコンビネーションによってコード0FCHを入力したい，というような場合はBIOSのエンコードファンクションをトラップするのが簡単でしょう．AH=8で，BLとALが指定の値である場合のみAXにxxFChを設定し復帰するようにするのです．それ以外の場合はデフォルトのエンコードルーチンに任せてしまうわけです．[ctrl]+ [graph]+ [del] でリセットをかけたい，というような場合も，同様なキーコンビネーションを監視すればよいでしょう．

最後に，これらに対応したプログラムをサンプルとしてリストに示すことにします．

それでは，創刊号から最終号までお付合いただき，ありがとうございました．そしてお疲れ様でした．

リスト2　ALTSHIFT.ASM

```
;=======================================
; altshift/ make shift key alternative
; link with res3
;=======================================
        .model  small, pascal

        include condx.inc
        include utils.inc
        masm51

        .stack  2000h

        .code
        public  my_id, interrupt_list, init_func, release_func
        public  check_func, intf_func

key_id  equ     258eh
my_id   dw      2590h

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

interrupt_list  label   byte
i18     int_work        <18h, xint18, 0>
        dw      0

shift_state     db      0

;――――――――――――――――――――
; program part
;――――――――――――――――――――
xint18:
        ; check function
        ; ah=9, bl=0ffh, al=70/f0h(shift code)
        cmp     bl, 0ffh
        @if     nz
                jmp     i18.int_old
        @endif
        push    ax
        and     al, not 80h
        cmp     ax, 0970h
        pop     ax
        @if     nz
                jmp     i18.int_old
        @endif

        ; check make bit
        test    al, 80h
        @if     nz
                ; alternate flag
                xor     shift_state, 1
        @endif

        ; check flag
        cmp     shift_state, 0
        @if     z
                ; ignore it
                mov     al, 0ffh
        @endif

        iret


release_func    proc
        ret
release_func    endp

intf_func       proc
        ret
intf_func       endp

init_seg        segment 'stack'
        assume  cs:init_seg

mes     db      'keybios is not installed', 0dh, 0ah, '$'

check_func      proc    far
        mov     ax, key_id
        mov     bx, 0
        mov     dx, 0
        int     2fh

        or      dx, dx
        @if     nz
                xor     ax, ax
        @else
                push    ds
                push    cs
                pop     ds
                mov     dx, offset mes
                mov     ah, 9
                int     21h

                xor     ax, ax
                inc     ax
        @endif
        ret
check_func      endp

init_func       proc    far
        ret
init_func       endp

init_seg        ends

        end

```

リスト3　NOKANA.ASM

```
;=======================================
; nokana/ kill kana key
; link with res3
;=======================================
        .model  small, pascal

        include condx.inc
        include utils.inc
        masm51

        .stack  2000h

        .code
        public  my_id, interrupt_list, init_func, release_func
        public  check_func, intf_func

key_id  equ     258eh
my_id   dw      2591h

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

interrupt_list  label   byte
        dw      0

        .data
kana_shift      db      ?
kana_pos        dd      ?

        .code
release_func    proc
        les     bx, kana_pos
        mov     al, kana_shift
        mov     es:[bx], al
        ret
release_func    endp

intf_func       proc
        ret
intf_func       endp

init_seg        segment 'stack'
        assume  cs:init_seg

mes     db      'keybios is not installed', 0dh, 0ah, '$'

check_func      proc    far
        mov     ax, key_id
        mov     bx, 0
        mov     dx, 0
        int     2fh

        or      dx, dx
        @if     nz
                xor     ax, ax
        @else
                push    ds
                push    cs
                pop     ds
                mov     dx, offset mes
                mov     ah, 9
                int     21h

                xor     ax, ax
                inc     ax
        @endif
        ret
check_func      endp

init_func       proc    far
        ; get internal parameter address
        mov     ah, 9
        mov     al, 0ffh
        mov     bl, 0ffh
        int     18h
        mov     es, ax
        add     bx, 2

        mov     word ptr kana_pos, bx
        mov     word ptr kana_pos+2, es
        mov     al, es:[bx]
        mov     kana_shift, al
        mov     byte ptr es:[bx], 0ffh

        ret
init_func       endp

init_seg        ends

        end

```


ワンラインNEWS
フジキンソフトはビデオ映像，写真，絵，CADデータをソニーのWS「NEWS」上で一括して管理できるソフト「画伯」を開発．INS64を使って遠隔地でも編集が可能．

短期集中連載

酒井法雄

WordBASICを使おう
―マクロの概要・応用編―

先月は，WordBASICとその母体であるMicrosoft Wordについて述べました．WordBASIC自体については，その言語仕様の概要をお知らせするにとどまりました．少々ものたりなかったかもしれません．今月は，実際にWordBASICを使ってプログラミングにチャレンジしてみます．

マクロの必要性

先月もご紹介したように，WordBASICはMicrosoft Windows上のワードプロセッサであるMicrosoft Wordのマクロです．マクロというと，低機能なものやキーボードマクロなどを思い浮かべますが，WordBASICはQuickBASICをWindows用に，そして文書処理用にカスタマイズしたような高機能なマクロ言語です．言語仕様の概要は先月にご紹介しましたので，今月は実際にこのWordBASICを使ってプログラミングしてみましょう．

さて，プログラミングとはいっても，WordBASICはワードプロセッサ用のマクロですから，かなり特殊なものであることは確かです．QuickBASICに似ているといったところで，言語仕様のほんの一部が似ているに過ぎません．やはり特殊な言語なのです．その特殊性を理解しないと，これからのプログラミングの方向性も見えてきません．そこで，まずは「なぜマクロが必要か」ということをもう一度考えてみましょう．

文書作成を能率的にするために

ワードプロセッサとは，その名のとおり言葉を加工するものです．実際には文章を書いたり，編集したり，最終的には出力したい形式に変更を加えたりといった作業をするわけです．その過程でもっとも重要な作業は，効率よく文章を作成し編集することでしょう．これらの作業のため，ワードプロセッサにはいろいろな機能があるわけです．

前回紹介したように，これらの機能の多くをMicrosoft Wordではマクロとして持っています．ユーザーは，これらのマクロを組み合わせて新しい

マクロを定義することができます．この定義をWordBASICで記述することができるわけです．では，どのようなことをユーザーは定義するのが本筋でしょうか．それは，当然ながら，ワードプロセッサの主たる目的である効率よく文章を作成し編集することのためでしょう．

効率よくということは，自動化につながります．すなわち，文書処理を人間の手をなるべく煩わせないようにすることがマクロの目的であるわけです．その究極には，思っただけで最適な表現の文章ができるということでしょうが，まだまだそこまではいきません．現状では，元の文章の用語を統一したり書式を統一するといったことが行われています．もっとも，思っただけで最適な文章（この最適というのがクセモノではありますが）になるのなら，文章は実に味気ないものに感じられるようになるでしょう．

いずれにせよ，速く美しい文章を書くことができるための補助手段として，マクロは存在しているといっていいでしょう．

マクロの方向性

それでは，私たちがこれからWordBASICのマクロを使って，どのようなことができるかをざっと考えてみましょう．

まず第一に考えられることは，用語の統一です．たとえば，「たとえば」と「例えば」という2つの用語をどちらかに統一することです．このような用語の不一致は文章を書く上でもっとも気をつけなくてはいけないことのひとつです．送り仮名の統一も必要でしょう．また，「ですます調」と「である調」などの統一も必要です．もっと基本的なところでは，2bytes文字と1bytes文字など文字種の統一も必要です．

次に考えられることは，まちがった文章を直すことです．これには，正しい文章であるリファレンスとしての辞書が必要となるでしょう．また，文法解析なども必要になるかもしれません．

さらに押し進めて，フォントなどの表現力を使いやすいものにするという方向性もあるでしょう．

もっとも，このような正道といえるようなものしかマクロにならないわけではありません．WordBASICのサンプルにも，文書作成とは無関係な，おもしろいマクロがあります．このようなものはWindowsという新しいOSで手軽にプログラミングできるWordBASICならではの強みであり，たいへん楽しいものですが，どうしてもそういうことがしたいのならば，Visual-BasicなりC＋SDKなりで作ったほうが正しく楽しいプログラミングができるでしょう．WordBASICは，やはり文書処理のための独自なプログラミングをしてこそ本領を発揮するものです．

文書の自動作成

マクロの方向性として，もっぱら文書の編集について述べましたが，WordBASICではネタさえあればもっと楽しい文書作成に関わるプログラミングができます．ここでは，編集の一歩前の作業をプログラミングしてみましょう．

編集の一歩前といえば，文章を書くことです．すなわち，ある程度自動的に文章を作成させてしまおうということです．

実は，私は物書きを職業としながらも，手紙を書くことが極めて苦手なのです．何が面倒かといえば，あの型にはまった文章です．どの順番で何を書くとか，時候の挨拶とか，頭と終りに何か書かないといけないとか……，考えただけで気が遠くなります．特に時候の挨拶というのは，何も思い付かないことが多く，手紙を書かない最大の原因のひとつとなっています（本当の最大の原因は面倒だからですが）．

ところが，手紙の書き方を書いた本を買ってきたら，その例文は，なんと決まり切った文章ばかりです．これなら，単文登録しておけばいいようなものです．ここで，ふと気付きました．単文登録しても，それを探すのが面倒です．それならいっそ，マクロにしてしまったらどうでしょうか．日付はWordBASICから知ることができますから，その月によって勝手に時候の挨拶くらいマクロに書かせることができそうです．

さらに，複数の挨拶から選択できるようにしておけば，文章の自由度も上がります．そこまでやるなら，ほかの決まり切った文章もマクロに書かせてしまったらどうでしょうか．しょせん，堅苦しい手紙を構成するパーツも文章です．それぞれのパートごとに，いくつかの文章から選択できるようにしておき，それぞれのパートを組み合わせれば，まとまった手紙にすることができるでしょう．

時候の挨拶を表示する

まずは，この企みを実現するための第一歩として，時候の挨拶を表示する簡単なプログラムを作ってみました．

時候の挨拶にもいろいろありますが，基本的にはその月によって変わります．本当は同じ月でも年によって気候は違うのですから，通り一遍の挨拶もヘンな話ではあります．しかし，これが堅苦しい挨拶なら，通り一遍でかまわないわけです．

ここでは，「○○の候，いかがお過ごしでしょうか.」というあまりにもありがちな堅苦しい文章を作ってみました．

リスト1（TEST1. BAS）をご覧ください．Date$()関数で取得した日付から，現在の月を取得し，その月に合わせた「○○の候」をSelect Case文で決定しています．これを元に文章を作り，Print文で出力しています．

ここで注意しなくてはいけないのは，通常のBASICならば「Date$」だけでいいところが，WordBASICでは「Date$()」とお尻に括弧が必要になることです．これを書かないとわけのわからないエラーになってしまいます．

どうやら，これはうまくいきそうです．しかし，ここで作ったプログラムは，図1のように，Wordの最下行に文章を表示するだけのものです．文章は，やはりちゃんと編集中のウィンドウに表示されるようにしたいところです．

データファイルを使う

文頭の時候の挨拶ができるのですから，当然ながら，「○○の折り，お身体にお気をつけて……」といった文末の時候の挨拶もできるはずです．しかし，こうなってくるとマクロ本体に書くのは大げさになってきます．

ここは，文章のデータは別ファイルとして持ち，プログラムの中で読み込むことにしましょう．もっとも，これではマクロ本体以外にファイルが必要になるわけですから，使い勝手が悪くなると思われる向きもあることでしょう．しかし，文章が別ファイルになっていれば，そのファイルを変更するだけで別の文章に変えることができるわけですから，かえってメンテナンス性は上がるわけです．

さて，ここでは1月から12月に対応する，文頭と文末の時候の挨拶を1つずつ「JIKOU. DAT」（リスト2）というファイルに書くことにしましょう．

リスト3（TEST2. BAS）では，このファイルをオープンして配列に読み込み，現在の月に対応する文章を

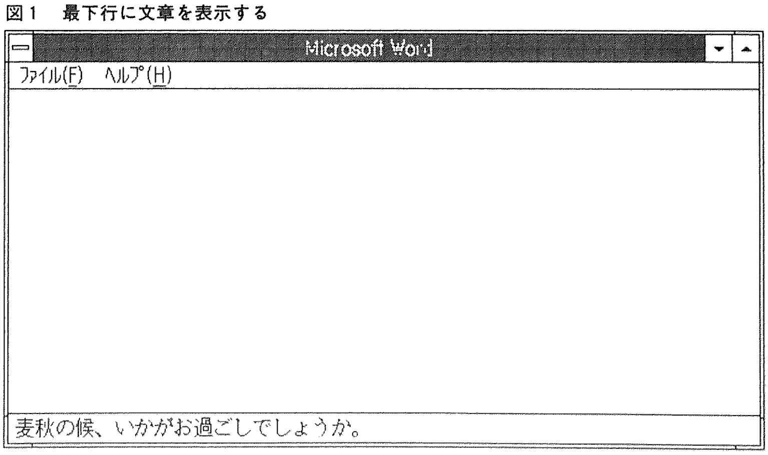
リスト1　TEST1.BAS，Select Caseを使って月別に時候の挨拶を挿入するマクロ

```
Sub MAIN
        month = Val(Mid$(Date$(), 4, 2))
        Select Case month
        Case 1
                message$ = "厳寒"
        Case 2
                message$ = "余寒"
        Case 3
                message$ = "早春"
        Case 4
                message$ = "春暖"
        Case 5
                message$ = "新緑"
        Case 6
                message$ = "麦秋"
        Case 7
                message$ = "盛夏"
        Case 8
                message$ = "晩夏"
        Case 9
                message$ = "初秋"
        Case 10
                message$ = "秋冷"
        Case 11
                message$ = "晩秋"
        Case 12
                message$ = "初冬"
        End Select
        Print message$ + "の候、いかがお過ごしでしょうか。
End Sub
```

リスト2　JIKOU.DAT

```
"厳寒","本格的な寒さはまだこれから、ご自愛のほどお祈り申し上げます。"
"余寒","余寒厳しき折りから、お風邪などお召しませんよう、お気を付けください。"
"早春","やさしい春の日におすこやかな毎日をお過ごしになられますよう。"
"春暖","春たけなわ、ご多忙をきわめる季節ですが、ご自愛のほど、お願い申しあげます。"
"新緑","木の芽どきはとかく疲れやすいものです。どうぞご自愛ください。"
"麦秋","時節がら、思わぬ風邪など召されませんよう、お気を付けください。"
"盛夏","暑さ厳しき折りから、くれぐれもお身体をお大切に。"
"晩夏","残暑厳しき折りから、どうかご無理をなさいませんようご自愛ください。"
"初秋","朝夕は冷え込む季節、お身体にはじゅうぶんお気をつけください。"
"秋冷","秋冷加わります折りから、ご自愛くださいませ。"
"晩秋","日一日と寒さに向いますが、お健やかにお過ごしください。"
"初冬","何かと気ぜわしい頃ですが、お身体にはくれぐれもお気をつけください。"
```

図1　最下行に文章を表示する



Insert文を使って編集中の文書に挿入するようにしています. Insert文は, カレントの注目位置に文字列を挿入するステートメントです.

ファイルからの読み込みは, Read文で行っています. 従来のBASICでのREAD文は, DATA文からデータを読み込むものでしたが, ここでは違っていることに注意してください. ここでのRead文は, 従来のBASICのLINE INPUT文に相当します.

プロシージャを使う

さて, これで大筋はOKのようです. このプログラムをどんどん拡大していって, 自動的に文章を作成するためには, その機能ごとにプロシージャ化する必要があります. そこで, ここではファイルからの読み込みルーチンをプロシージャの形にしましょう.

リスト4 (JIKOU1. BAS) では, 文頭と文末の時候の挨拶を12か月分格納する文字列型変数の配列をShared宣言して, グローバル変数としています. グローバル変数の宣言は, MAINを含むいずれのプロシージャの中にあってもいけません.

また, QuickBASICではプロシージャの引数に配列を渡したいとき, リスト5のように書けば, 配列も引数で渡すことができましたが, WordBASICでは残念ながらこれができません. したがって, グローバル変数にするしかありません.

ともあれ, WordBASICの文法や機能を使って文章の自動化をできる見込みは十分につきました. 次は, さらに自由度を増す方向性で考えてみましょう.

ダイアログボックスを使う……オプションボタン

さきほどのマクロを使えば, その月に合わせた時候の挨拶が自動生成されます. しかし, 手紙を書くのが月末ならば, 出すのは翌月になることもあるでしょう. また, あらかじめ先のことを書くこともあるでしょう. こうなると, Date$()から自動的に時候の挨拶ができるのも良し悪しです.

そこで, ダイアログボックスを使い, オプションボタンで選んだ月に対応した時候の挨拶が生成されるようにしてみましょう.

リスト6 (JIKOU2. BAS) では, Date$()から取得した月をデフォルトとして, 12個のオプションボタンからひとつだけを選択できるようにしたダイアログボックスを追加しています. ここで, ダイアログボックスの使い方について, あらためて述べます.

1.ダイアログボックスの定義

「Begin Dialog UserDialog」と「End Dialog」間にダイアログボックスの定義を記述します. このとき, BeginDialogの後は必ず「UserDialog」と記述しなくてはいけません. 前回の例ではまちがった記述をしてありました. お詫びして訂正いたします.

2.オプションボタンの定義

「GroupBox」のエリア内にあるオプションボタンは, そのうちひとつしか選択できません. ここでは, すべてのオプションボタンをひとつのグループにしてあります. また, 「Option-

リスト3　TEST2.BAS, ファイル化したデータを読み込んでから処理するマクロ

```
Sub MAIN
        Dim bjikou$(13), ejikou$(13)

        Open "jikou.dat" For Input As #1
        While(i < 12)
                i = i + 1
                Read #1, bjikou$(i), ejikou$(i)
        Wend
        Close #1
        month = Val(Mid$(Date$(), 4, 2))
        Insert bjikou$(month) + "の候、いかがお過ごしでしょうか。"
        Insert ejikou$(month)
End Sub
```

リスト4　JIKOU1.BAS, 読み込みプロシージャを別にしたマクロ

```
Dim Shared bjikou$(13), ejikou$(13)
Sub MAIN
        cr$ = Chr$(13) : sp$ = "　"
        LoadJikou
        month = Val(Mid$(Date$(), 4, 2))
        Insert sp$ + bjikou$(month) + "の候、いかがお過ごしでしょうか。" +
cr$
        Insert sp$ + ejikou$(month) + cr$
End Sub

'
' 時候の挨拶データ読み込み
'
Sub LoadJikou
        Open "jikou.dat" For Input As #1
        While(i < 12)
                i = i + 1
                Read #1, bjikou$(i), ejikou$(i)
        Wend
        Close #1
End Sub


```

リスト5

```
Sub LoadJikou (bjikou$())
```

Group.month」は，グループ内のオプションボタンのデフォルトや選択されたものの順番を表すための定義です．この順番は，記述した順に0から始まる数値でアクセスできます．ここでは，「month」というメンバー名でアクセスすることになります．

3.ダイアログボックスの宣言

定義したダイアログボックスを実際に使う前に，ダイアログボックスとして使われる変数エリアを宣言しなくてはいけません．「Dim Dlg As Dialog UserDialog」では，Dlgというダイアログ変数を宣言しています．

4.デフォルトの設定

ダイアログを表示する前に，オプションボタンのデフォルト値を設定します．ここでは，dlgというダイアログ変数でmonthというグループを表すメンバー名を使っていますから，「dlg.month」でアクセスすることができます．

リスト7では，月を表す数値より1少なく設定しています．これは，グループ内の順番は0から始まるので，1月なら0，6月なら5を設定しなくてはいけないからです．

5.ダイアログボックスの表示

「Dialog dlg」で，ダイアログが表示されます．

6.選択されたグループの戻り値

デフォルトの設定と同じように，「dlg.month」に選択された戻り値が設定されます．これも0からの並びですから，「month = dlg.month + 1」のように1を加えて月を表す数値にしています．

このようにして作成したダイアログは，図2のように表示されます．

実際にやってみると，プログラミング自体よりもダイアログのデザインが面倒なことがわかります．また，文書ファイルをアクティブにしておいた状態でマクロを実行しないと，途中でエラーになってしまいます．かといって，文書ファイルを最大化した状態のままではデバッグの切り替えがしづらいですから，狭くなりますが文書ファイルとマクロファイルが同時に表示されるようにしてデバッグするのがいいでしょう．

リスト6　JIKOU2.BAS，月の指定をダイアログでできるようにしたマクロ．デフォルトあり

```
Dim Shared bjikou$(13), ejikou$(13), cr$, sp$

Sub MAIN
        Begin Dialog UserDialog 300, 220
                Text 10, 8, 250, 18, "手紙を送る月を選んでください。"
                GroupBox 10, 20, 200, 150, "月"
                OptionGroup .month
                OptionButton 15, 30, 110, 18, "1月"
                OptionButton 15, 50, 50, 18, "2月"
                OptionButton 15, 70, 50, 18, "3月"
                OptionButton 15, 90, 50, 18, "4月"
                OptionButton 15, 110, 50, 18, "5月"
                OptionButton 15, 130, 50, 18, "6月"
                OptionButton 100, 30, 50, 18, "7月"
                OptionButton 100, 50, 50, 18, "8月"
                OptionButton 100, 70, 50, 18, "9月"
                OptionButton 100, 90, 50, 18, "10月"
                OptionButton 100, 110, 50, 18, "11月"
                OptionButton 100, 130, 50, 18, "12月"
                OKButton 105, 180, 88, 18
        End Dialog

        Dim Dlg As Dialog UserDialog
        cr$ = Chr$(13) : sp$ = "　"
        LoadJikou

        dlg.month = Val(Mid$(Date$(), 4, 2)) - 1
        Dialog dlg
        month = dlg.month + 1
        Insert sp$ + bjikou$(month) + "の候、いかがお過ごしでしょうか。" +
cr$
        Insert sp$ + ejikou$(month) + cr$
End Sub
'
' 時候の挨拶データ読み込み
'
Sub LoadJikou
        Open "jikou.dat" For Input As #1
        While(i < 12)
                i = i + 1
                Read #1, bjikou$(i), ejikou$(i)
        Wend
        Close #1
End Sub
```

リスト7

```
dlg.month = Val(Mid$(Date$(), 4, 2)) - 1
```

ダイアログボックスを使う……リストボックス

次に，複数の時候の挨拶を選択できるようにしてみましょう．データの形式が変わります．ここでは，頭の時候の挨拶のデータとしてリスト8（JIKOU_B.DAT），終りの時候の挨拶のデータとしてリスト9（JIKOU_E.DAT）の2つのファイルを用意しました．

このプログラム（リスト10，JIKOU3.BAS）では，月を選択するダイアログのプロシージャとしてGetMonth()，時候の挨拶を選択するGetJikou$(month, DataName$)の2つを用意しました．それぞれのプロシージャは戻り値のあるFunctionです．このようにプロシージャにしておけば，後から使い回しができて便利です．

リストボックスから時候の挨拶を選

択するGetJikou$について解説しましょう．リストボックスを表示するためのダイアログボックスの定義を見ると，リスト12のようになっています．

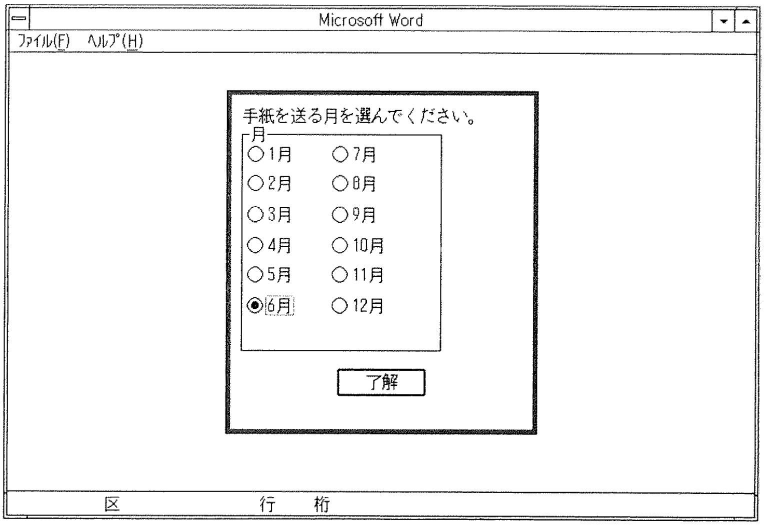
「jikou$()」は，リストボックスに表示される文字列の配列です．「.jikouno」は，リストボックスのアイテムの番号を示す数値です．デフォルト値と戻り値は，「dlg.jikouno」のようにしてアクセスすることができます．グループのときとよく似ています．

こうしてできたプログラムには，少々問題がありました．というのは，WordBASICでのファイルの読み込みは非常に遅いのです．複数件の時候データを読み込むのにも，ずいぶんと時間がかかってしまいます．シーケンシャルファイルであることを加味しても，ちょっと遅すぎます．そこで，ここでは該当する月データを読み込んだら，読み込みループを脱出するように書きました．それでも，12月分のデータ読み込みには，めいっぱい時間がかかってしまいます．これはいたしかたありません．

また，当初はGetMonthだけをプロシージャにし，メインで時候の挨拶を選択するダイアログボックスを表示させようとしたところ，GetMonthのときのダイアログボックスが2度表示されてしまいました．今の形にしたところ，正常に動作するようになりました．詳細は不明ですが，どうもWordBASICのバグではないかと思われます．

これで，月を指定して，複数の例か

図2　ダイアログが表示される



リスト8　JIKOU_B.DAT

```
1,"厳寒の候"
1,"酷寒のみぎり"
1,"寒に入り、寒さも一段と厳しくなった今日このごろ"
1,"本年は格段の寒さですが"
1,"連日の寒さに春が待ち遠しい今日このごろ"
1,"新年を迎え"
2,"余寒の候"
2,"余寒なお厳しい折り"
2,"春寒厳しき折りから"
2,"立春とは名ばかり"
2,"春まだ浅く"
2,"三寒四温とは申しますが"
2,"梅のつぼみがふくれはじめました今日このごろ"
2,"もうすぐバレンタインデーですが"
2,"バレンタインデーの収穫も済み"
2,"いよいよ受験のシーズンとなりまたが"
3,"早春の候"
3,"弥生三月"
3,"寒さも緩み"
3,"春色にわかに動き始めた今日このごろ"
3,"日に日に春めいてきましたが"
3,"ちらほら春の便りも聞かれるこのごろ"
3,"暑さ寒さも彼岸までとはよく申したものですが"
3,"いよいよ卒業のシーズンとなりましたが"
4,"春暖の候"
4,"春陽麗和のみぎり"
4,"桜花の季節"
4,"春もたけなわとなりましたが"
4,"花曇りのおだやかな今日このごろ"
4,"桜も満開となりましたが"
4,"新年度を迎え"
4,"入学式のシーズンとなりましたが"
4,"ゴールデンウィーク進行の季節"
5,"新緑の候"
5,"風薫る五月"
5,"晩春のみぎり"
5,"若葉のさわやかな季節"
5,"待ちに待ったゴールデンウィーク"
5,"春眠暁を覚えずと申しますが"
5,"すっかり夏めいて汗ばむほどの陽気になりましたが"
5,"五月病の季節となりましたが"
6,"麦秋の候"
6,"木々の緑も濃さを増すころ"
6,"衣更えの季節"
6,"女子高生の制服も白くなった今日このごろ"
6,"うっとうしい梅雨に入りましたが"
6,"梅雨冷えの折り"
6,"長雨に気分も沈みがちになりますが"
6,"雨に洗われた緑が美しい今日このごろ"
6,"梅雨の晴れ間のすがすがしさ"
6,"ジューンブライドと申しますが、今年も関係なく"
7,"盛夏の候"
7,"炎暑の候"
7,"猛夏のみぎり"
7,"梅雨明けの暑さもひとしお"
7,"夏もたけなわ"
7,"猛暑の季節"
7,"緑陰の恋しい季節となりましたが"
7,"もうすぐ夏休みですが"
7,"いよいよハイレグの季節となりましたが"
8,"晩夏の候"
8,"残暑の候"
8,"暮夏のみぎり"
8,"土用のあとの暑さ、ことのほか厳しく"
8,"暑さももうしばらく"
8,"朝夕はすっかり秋の気配となりましたが"
8,"いよいよ夏休みも終りですが"
9,"初秋の候"
9,"新秋の候"
9,"爽涼のみぎり"
9,"灯火したしむころ"
9,"一雨ごとに秋色加わり"
9,"いよいよ新学期ですが"
9,"夏休みの宿題の残りもいまだ多い今日このごろ"
10,"秋冷の候"
10,"爽秋のみぎり"
10,"秋色深みゆくこのごろ"
10,"スポーツの秋"
10,"サンマも美味しくなりましたが"
10,"秋の夜長"
10,"庭木も色づきはじめたこのごろ"
10,"朝夕は肌寒いほどですが"
10,"読書の秋となりましたが"
10,"一人身が身にしみる季節となりましたが"
11,"晩秋の候"
11,"向寒のみぎり"
11,"秋色いよいよ深く"
11,"朝夕の寒さが身にしみる季節"
11,"日に日に冷気が加わります折り"
11,"木々も葉を落す今日このごろ"
11,"立秋を迎え"
11,"寒い冬がもうそこまでやってきておりますが"
11,"一人身が寂しい今日このごろ"
11,"年末進行の季節"
12,"初冬の候"
12,"寒冷の候"
12,"師走の声を聞き"
12,"寒気ますます厳しいおり"
12,"寒さもひとしお身にしみるころ"
12,"街にはジングルベルの曲が流れていますが"
12,"年の瀬もせまり"
12,"いよいよ今年も押し詰まり"
12,"今年もティファニーとホテルの儲けるシーズンがきましたが"
```


ワンラインNEWS
日本情報処理開発協会は，第1種情報処理技術者試験と第2種情報処理技術者試験の試験科目のうち，プログラム作成能力の科目に「C言語」を選択言語に追加する．

ら時候の挨拶を選択し，文書に挿入するプログラムができました．ここまでできてしまえば，こっちのものです．いよいよ，手紙自体を書いてしまうプログラムを作ってみることにしましょう．

定型文書の大枠を作るプログラム

手紙には，ややこしい順番があります．一般的には，頭語，前文，主文，末文，結語，日付，署名，宛名，脇付，副文といった順番に書くようです．主文と宛名，日付といったもの以外は，ほとんど決まりきったものですから，これもプログラムに書かせてしまいましょう．JIKOU3. BASを少々改造して，問題のデータさえ作ってしまえばできるはずです．

よりイカすプログラミングを考えながら作ったのが，リスト13 (LETTER. BAS) です．このプログラムのために，JIKOU_B. DAT，JIKOU_C. DAT, JIKOU_D. DAT, JIKOU_E. DAT, ATAMA. DAT, KETUGO. DAT, MUSUBI. DAT, TUNAGI. DAT，WAKITUKE. DATとたくさんのデータファイルを用意しました(リスト8，14，15，9，16，17，18，19, 20)．時候の挨拶もより細かく設定することができるようになっています．それぞれのデータは通常のテキストファイルですから，自由にカスタマイズすることができます．また，文章の位置なども適切な位置に書かれるようになっています．

このマクロで一通りの文章が完成し

リスト9　JIKOU_E.DAT

```
1,"本格的な寒さはまだこれから、ご自愛のほどお祈り申し上げます。"
1,"今年もよい年でありますよう、お祈り申し上げております。"
1,"寒さ厳しき折りから、くれぐれもお身体を大切にしてください。"
1,"今年こそ入学できるよう、影ながらお祈り申し上げております。"
1,"今年こそご結婚できるよう、影ながらお祈り申し上げております。"
2,"余寒厳しき折りから、お風邪などお召しませんよう、お気を付けください。"
2,"余寒は身にしみます。くれぐれもお身体をおいといくださいますよう、お願い申し上げます。"
2,"寒さももう一息、お身体をお大事に。"
2,"バレンタインの成果など気にせず、がんばってください。"
3,"やさしい春の日におすこやかな毎日をお過ごしになられますよう。"
3,"春とはいえ、まだ寒い日もございますから、お身体を大切に。"
3,"お花見のさいには、おきそいください。"
3,"来年こそ晴れやかな気分で桜を見れますよう。"
3,"ホワイトデーのお返しはお忘れなく。"
4,"春たけなわ、ご多忙をきわめる季節ですが、ご自愛のほど、お願い申しあげます。"
4,"この花の季節をこころゆくまでお楽しみください。"
4,"しばらくはもっともよい季節です。ますますお元気でお過ごしください。"
4,"新入生に手をだすのもほどほどにして、お健やかに。"
5,"木の芽どきはとかく疲れやすいものです。どうぞご自愛ください。"
5,"季節の変わりめですので、お身体を大切に。"
5,"新緑のようにすがすがしい季節をお送りくださいますよう。"
5,"5月病などならないよう、お健やかにお過ごしください。"
6,"時節がら、思わぬ風邪など召されませんよう、お気を付けください。"
6,"うっとうしい毎日、ご無理をなさいませんよう。"
6,"梅雨明けも間近いようです。ご自愛専一のほど、お祈り申し上げます。"
6,"来年の6月こそ花嫁となれるようお祈り申し上げます。"
7,"暑さ厳しき折りから、くれぐれもお身体をお大切に。"
7,"夏風邪など召しませんよう、お気をつけください。"
7,"楽しい夏休みをお過ごしください。"
7,"夏休みなどなくても、仕事に生きてこそ男です。徹夜に明け暮れてください。"
8,"残暑厳しき折りから、どうかご無理をなさいませんようご自愛ください。"
8,"そろそろ夏の疲れが出るころ、お身体を大切になさってください。"
8,"実りの秋に向かって、ますますのご活躍をお祈り申し上げます。"
8,"夏休みもすぐに終ります。遊びすぎないよう、宿題もしてください。"
9,"朝夕は冷え込む季節、お身体にはじゅうぶんお気をつけください。"
9,"夏の疲れが出るころですから、お身体には充分お気をつけてください。"
9,"季節の変わり目です。お身体を大切に。"
9,"徹夜にはいい季節です。原稿がんばってください。"
10,"秋冷加わります折りから、ご自愛くださいませ。"
10,"夜ともなるとめっきり冷えてまいります。どうぞお身体を大切にあそばしますよう。"
10,"快適な季節、お健やかにお過ごしください。"
10,"秋の夜長、読書のしすぎにはご注意ください。"
10,"食欲の秋とはいえ、食べるのもほどほどに。"
11,"日一日と寒さに向いますが、お健やかにお過ごしください。"
11,"向寒の折りから、ご健康にはどうぞご注意ください。"
11,"近付く冬にそなえて、お身体を大切に。"
11,"ギンナンの食べすぎにはご注意ください。"
12,"何かと気ぜわしい頃ですが、お身体にはくれぐれもお気をつけください。"
12,"寒さ厳しき折りから、ご自愛のほどお祈り申し上げます。"
12,"きたるべき年もあなたにとってよいお年でありますよう。"
12,"クリスマスに向けて、アルバイトがんばってください。"
12,"押し迫ったとはいえ、今年もまだ少しはあります。がんばってください。"
```

リスト10　JIKOU3.BAS，複数の時候の挨拶をダイアログのリストボックスから選択できるようにしたマクロ

```
Dim Shared CR$, SP$, TRUE, FALSE
Dim Shared jikou$(20)

Sub MAIN
	CR$ = Chr$(13) : SP$ = "　" : TRUE = - 1 : FALSE = 0
	month = GetMonth
	jikoub$ = GetJikou$(month, "JIKOU_B.DAT")
	Insert SP$ + jikoub$ + "、いかがお過ごしでしょうか。" + CR$
	jikoue$ = GetJikou$(month, "JIKOU_E.DAT")
	Insert SP$ + jikoue$ + CR$
End Sub
'
' 時候の挨拶の選択
'
Function GetJikou$(month, DataName$)
	For i = 0 To 19
		jikou$(i) = ""
	Next i
	LoadJikou(month, DataName$)
	Begin Dialog UserDialog 500, 200
		Text 50, 8, 250, 18, "時候の挨拶を選んでください。"
		ListBox 15, 25, 470, 140, jikou$(), .jikouno
		OKButton 120, 170, 88, 18
	End Dialog
	Dim Dlg As Dialog UserDialog

	Dialog dlg
	GetJikou$ = jikou$(dlg.jikouno)
End Function
'
' 月を得る
'
Function GetMonth
	Begin Dialog UserDialog 500, 80
		Text 50, 8, 250, 18, "時候の月を選んでください。"
		GroupBox 10, 20, 350, 50, "月"
		OptionGroup .month
		OptionButton 15, 30, 50, 18, "1月"
		OptionButton 70, 30, 50, 18, "2月"
		OptionButton 125, 30, 50, 18, "3月"
		OptionButton 180, 30, 50, 18, "4月"
		OptionButton 235, 30, 50, 18, "5月"
		OptionButton 290, 30, 50, 18, "6月"
		OptionButton 15, 50, 50, 18, "7月"
		OptionButton 70, 50, 50, 18, "8月"
		OptionButton 125, 50, 50, 18, "9月"
		OptionButton 180, 50, 50, 18, "10月"
		OptionButton 235, 50, 50, 18, "11月"
		OptionButton 290, 50, 50, 18, "12月"
		OKButton 380, 30, 88, 18
	End Dialog

	Dim Dlg As Dialog UserDialog
	dlg.month = Val(Mid$(Date$(), 4, 2)) - 1
	Dialog dlg
	GetMonth = dlg.month + 1
End Function
'
' 時候の挨拶データ読み込み
'
Sub LoadJikou(month, DataName$)
	i = 0
	quit = FALSE
	Open DataName$ For Input As #1
	While(Not(Eof(1) Or quit = TRUE))
		Read #1, dum, dummy$
		If dum = month Then
			jikou$(i) = dummy$
			i = i + 1
		ElseIf dum = month + 1 Then
			quit = TRUE
		End If
	Wend
	Close #1
End Sub
```

リスト11 JIKOU1.DAT

```
1,"厳寒の候"
酷寒のみぎり
寒に入り、寒さも一段と厳しくなった今日このごろ
本年は格段の寒さですが
連日の寒さに春が待ち遠しい今日このごろ
新年を迎え
2,"余寒の候"
余寒なお厳しい折り
春寒厳しき折りから
立春とは名ばかり
春まだ浅く
三寒四温とは申しますが
梅のつぼみがふくれはじめました今日このごろ
もうすぐバレンタインデーですが
バレンタインデーの収穫も済み
いよいよ受験のシーズンとなりまたが
3,"早春の候"
弥生三月
寒さも緩み
春色にわかに動き始めた今日このごろ
日に日に春めいてきましたが
ちらほら春の便りも聞かれるこのごろ
暑さ寒さも彼岸まではよく申したものですが
いよいよ卒業のシーズンとなりましたが
4,"春暖の候"
春陽麗和のみぎり
桜花の季節
春もたけなわとなりましたが
花曇りのおだやかな今日このごろ
桜も満開となりましたが
新年度を迎え
入学式のシーズンとなりましたが
ゴールデンウィーク進行の季節
5,"新緑の候"
風薫る五月
晩春のみぎり
若葉のさわやかな季節
待ちに待ったゴールデンウィーク
春眠暁を覚えずと申しますが
すっかり夏めいて汗ばむほどの陽気になりましたが
五月病の季節となりましたが
6,"麦秋の候"
木々の緑も濃さを増すころ
衣更えの季節
女子高生の制服も白くなった今日このごろ
うっとうしい梅雨に入りましたが
梅雨冷えの折り
長雨に気分も沈みがちになりますが
雨に洗われた緑が美しい今日このごろ
梅雨の晴れ間のすがすがしさ
ジューンブライドと申しますが、今年も関係なく
7,"盛夏の候"
炎暑の候
猛夏のみぎり
梅雨明けの暑さもひとしお
夏もたけなわ
猛暑の季節
緑陰の恋しい季節となりましたが
もうすぐ夏休みですが
いよいよハイレグの季節となりましたが
8,"晩夏の候"
残暑の候
暮夏のみぎり
土用のあとの暑さ、ことのほか厳しく
暑さももうしばらく
朝夕はすっかり秋の気配となりましたが
いよいよ夏休みも終りですが
9,"初秋の候"
新秋の候
爽涼のみぎり
灯火したしむころ
一雨ごとに秋色加わり
いよいよ新学期ですが
夏休みの宿題の残りもいまだ多い今日このごろ
10,"秋冷の候"
爽秋のみぎり
秋色深みゆくこのごろ
スポーツの秋
サンマも美味しくなりましたが
秋の夜長
庭木も色づきはじめたこのごろ
朝夕は肌寒いほどですが
読書の秋となりましたが
一人身が身にしみる季節となりましたが
11,"晩秋の候"
向寒のみぎり
秋色いよいよ深く
朝夕の寒さが身にしみる季節
日に日に冷気が加わります折り
木々も葉を落す今日このごろ
立秋を迎え
寒い冬がもうそこまでやってきておりますが
一人身が寂しい今日このごろ
年末進行の季節
12,"初冬の候"
寒冷の候
師走の声を聞き
寒気ますます厳しいおり
寒さもひとしお身にしみるころ
街にはジングルベルの曲が流れていますが
年の瀬もせまり
いよいよ今年も押し詰まり
今年もティファニーとホテルの儲けるシーズンがきましたが


1,"本格的な寒さはまだこれから、ご自愛のほどお祈り申し上げます。"
2,"余寒厳しき折りから、お風邪などお召しませんよう、お気を付けください。"
3,"やさしい春の日におすこやかな毎日をお過ごしになられますよう。"
4,"春たけなわ、ご多忙をきわめる季節ですが、ご自愛のほど、お願い申しあげます。"
5,"木の芽どきはとかく疲れやすいものです。どうぞご自愛ください。"
6,"時節がら、思わぬ風邪など召されませんよう、お気を付けください。"
7,"暑さ厳しき折りから、くれぐれもお身体をお大切に。"
8,"残暑厳しき折りから、どうかご無理をなさいませんようご自愛ください。"
9,"朝夕は冷え込む季節、お身体にはじゅうぶんお気をつけください。"
10,"秋冷加わります折りから、ご自愛くださいませ。"
11,"日一日と寒さに向いますが、お健やかにお過ごしください。"
12,"何かと気ぜわしい頃ですが、お身体にはくれぐれもお気をつけください。"
```

リスト12

```
ListBox 15, 25, 470, 140, jikou$(), .jikouno
```

リスト13 LETTER.BAS

```
Dim Shared CR$, SP$, TAB$, TRUE, FALSE, MAXITEM
Dim Shared item$(20)

Sub MAIN
        '変数の初期化
        CR$ = Chr$(13) : SP$ = "  " : TAB$ = Chr$(9)
        TRUE = - 1 : FALSE = 0 : MAXITEM = 19

        Insert GetItem$("ATAMA.DAT", "頭語を選んでください。") + CR$ + CR$
        month = GetMonth
        Insert SP$ + GetJikou$(month, "JIKOU_B.DAT")
        Insert GetItem$("JIKOU_C.DAT", "時候の挨拶の後半の文章を選んでください。")
        Insert GetItem$("JIKOU_D.DAT", "前文の文章を選んでください。") + CR$
        Insert SP$ + GetItem$("TUNAGI.DAT", "つなぎの言葉を選んでください。")
        InsertBookmark "Main"
        Insert CR$ + CR$ + CR$
        Insert SP$ + GetJikou$(month, "JIKOU_E.DAT")
        Insert GetItem$("MUSUBI.DAT", "結びの文章を選んでください。") + CR$
        Insert TAB$ + TAB$ + TAB$ + GetItem$("KETUGO.DAT", "結語を選んでください。") + CR$
        yy$ = Str$(Val(Left$(Date$(), 2)) - 88)
        mm$ = Str$(month)
        dd$ = Mid$(Date$(), 7, 2)
        Insert TAB$ + "平成" + yy$ + "年" + mm$ + "月" + dd$ + "日" + CR$
        Insert TAB$ + TAB$ + TAB$ + TAB$ + "酒井 法雄" + CR$
        Insert SP$ + "様" + CR$
        Insert TAB$ + GetItem$("WAKITUKE.DAT", "脇付を選んでください。") + CR$
        Insert TAB$ + TAB$ + "追伸" + CR$
        EditGoTo "Main"
End Sub
'
' アイテムデータの選択
'
Function GetItem$(DataName$, Message$)
        LoadItem(DataName$)
        Begin Dialog UserDialog 500, 200
                Text 50, 8, 450, 18, Message$
                ListBox 15, 25, 470, 140, item$(), .itemno
                OKButton 120, 170, 88, 18
        End Dialog
        Dim Dlg As Dialog UserDialog

        Dialog dlg
        GetItem$ = item$(dlg.itemno)
End Function
'
' 時候の挨拶の選択
'
Function GetJikou$(month, DataName$)
        LoadJikou(month, DataName$)
        Begin Dialog UserDialog 500, 200
                Text 50, 8, 450, 18, "時候の挨拶を選んでください。"
                ListBox 15, 25, 470, 140, item$(), .jikouno
                OKButton 120, 170, 88, 18
        End Dialog
        Dim Dlg As Dialog UserDialog

        Dialog dlg
        GetJikou$ = item$(dlg.jikouno)
End Function
'
' 月を得る
'
Function GetMonth
        Begin Dialog UserDialog 500, 80
                Text 50, 8, 250, 18, "時候の月を選んでください。"
                GroupBox 10, 20, 350, 50, "月"
                OptionGroup .month
                OptionButton 15, 30, 50, 18, "1月"
                OptionButton 70, 30, 50, 18, "2月"
                OptionButton 125, 30, 50, 18, "3月"
                OptionButton 180, 30, 50, 18, "4月"
                OptionButton 235, 30, 50, 18, "5月"
                OptionButton 290, 30, 50, 18, "6月"
                OptionButton 15, 50, 50, 18, "7月"
                OptionButton 70, 50, 50, 18, "8月"
                OptionButton 125, 50, 50, 18, "9月"
                OptionButton 180, 50, 50, 18, "10月"
                OptionButton 235, 50, 50, 18, "11月"
                OptionButton 290, 50, 50, 18, "12月"
                OKButton 380, 30, 88, 18
        End Dialog

        Dim Dlg As Dialog UserDialog
```


ワンラインNEWS
米サイリックスはインテルの 32 ビットMPU「i 486」との互換製品を発売. 同社はすでに「i 386」との互換製品を発売しており, 両社間の特許紛争に発展している.

```
        dlg.month = Val(Mid$(Date$(), 4, 2)) - 1
        Dialog dlg
        GetMonth = dlg.month + 1
End Function
'
' 時候の挨拶データ読み込み
'
Sub LoadJikou(month, DataName$)
        For i = 0 To MAXITEM
                item$(i) = ""
        Next i
        i = 0
        quit = FALSE
        Open DataName$ For Input As #1
        While(Not(Eof(1) Or quit = TRUE))
                Read #1, dum, dummy$
                If dum = month Then
                        item$(i) = dummy$
                        i = i + 1
                ElseIf dum = month + 1 Then
                        quit = TRUE
                End If
        Wend
        Close #1
End Sub
'
' アイテムデータ読み込み
'
Sub LoadItem(DataName$)
        For i = 0 To MAXITEM
                item$(i) = ""
        Next i
        i = 0
        Open DataName$ For Input As #1
        While(Not Eof(1))
                Read #1, item$(i)
                i = i + 1
        Wend
        Close #1
End Sub
```

たら，本文の位置にカーソルが移動するようにしてあります．ここでは，「InsertBookmark "Main"」のように「Main」という名前のブックマークを設定しておき，あとから「EditGoTo "Main"」で，そのブックマークにカーソル位置を変更しています．このように，ブックマークはいろいろな場面で役立つことでしょう．

LETTERマクロで作成した文章の例が，SAMPLE1.DOC(図3)です．これに本文を追加したのが，SAMPLE2.DOC(図4)です．文章の内容には他意はありませんが，いつも私はこのように遅筆で編集部の皆様に御迷惑をかけております．本当に申しわけなく思っています．これをフォントの変更やアンダーラインで修飾したのが，SAMPLE3.DOC(図5)です．内容はともかく，ずいぶんと簡単にそれっぽくなります．

このようにして，文書処理を自動化していけば，文章を書くことが苦手な方でも楽しいワードプロセッシングができそうです．

複数文字列の一括置換

文書処理といえば，用語の統一はもっとも重要なテーマのひとつです．用語の統一には，もっぱら文字列の変換をするわけですが，複数の文字列を一括して変換できるとこの作業は楽になります．従来，UNIXやMS-DOSでは，sedというストリームエディタでこの作業をすることが多かったようです．また，複雑な変換には，awkやperlなどが使われました．これらの作業をWordBASICにやらせることは，正規表現以外のことについては問題なくできます．

ここでは，あまり複雑なことはせず，単に別ファイル(リスト21，REPLACE.DAT)に書かれた通りに文字列を一括して変換するプログラム(リスト22，REPLACE.BAS)を作ってみました．このプログラムでの第一のポイントは，文書の頭を示すブックマーク

リスト14　JIKOU_C.DAT

```
"、いかがお過ごしでしょうか。"
"、その後いかがお過ごしでしょうか。"
"、その後いかがですか。"
"、皆様にはお変わりないことと存じます。"
"、皆様にはお変わりなくご健勝のことと存じます。"
"、皆様にはお変わりなくご健勝のこととお慶び申し上げます。"
"、お元気のことと存じます。"
"、お元気ですか。"
"、お元気でしょうか。"
"、すっかり御無沙汰しておりますが、皆様にはお変わりないことと存じます。"
"、御無沙汰しておりますが、皆様にはお変わりなくご健勝のことと存じます。"
"、御無沙汰しておりますが、皆様にはお変わりなくご健勝のこととお慶び申し上げます。"
"、御無沙汰しておりますが、お元気のことと存じます。"
"、御無沙汰しておりますが、お元気ですか。"
"、御無沙汰しておりますが、お元気でしょうか。"
```

リスト15　JIKOU_D.DAT

```
こちらは皆、元気です。
おかげさまで、こちらは皆、元気です。
私は、元気です。
おかげさまで、私は元気です。
私は相変わらず元気です。
私は風邪をひいてしまったのですが、次第によくなってきています。
私は相変わらず忙しい毎日ですが、おかげさまで元気です。
先日はすっかりお世話になりまして、ありがとうございました。
先日は……していただき、まことにありがとうございました。
いつもお世話になっております。
```

リスト16　ATAMA.DAT

```
拝啓
拝呈
啓上
謹啓
一筆申し上げます。
謹んで申し上げます。
急拝
急呈
取り急ぎ申し上げます。
初めてお手紙を差し上げます。
突然のお手紙お許しください。
拝復
謹復
お手紙拝見いたしました。
取り急ぎご返事申し上げます。
```

「¥StartOfDoc」に移動することです．実際の変更は，「EditReplace」を使えば，Wordから文字列の置換をするときに指定する，大文字小文字の区別などのオプションをそのまま指定することができます．

ここでのデータファイルは，「ですます調」の文章を「である調」に変えるというものです．もっとも，サンプルとして作った不完全なデータですから，きちんと変更されません．

このマクロとデータを使ってSAMPLE3. DOCを変更したのが，SAMPLE4. DOC（図6）です．もともと失礼な文章だったのが，さらに腹立たしいほど失礼な文章になりました．語尾だけで，ずいぶんと感じが変わるものです．このように，用語の統一をマクロですれば，文書編集にはかなりの影響力を持つことができるでしょう．

自動文書作成プログラム

WordBASICには，ほかのBASICと同様に「Rnd」というランダムな数値を出す関数があります．これをうまく使えば，ゲームなどに応用することもできます．ゲームといっても，WordBASICの特徴を生かすような，文書とからめたものがおもしろいと思います．

ここでは，とりあえず自動的に文章を書くプログラム（リスト23,

図3　SAMPLE1.DOC，letterマクロで作成

```
突然のお手紙お許しください。

　梅雨明けの暑さもひとしお、皆様にはお変わりなくご健勝のことと存じます。
私は風邪をひいてしまったのですが、次第によくなってきています。
　早速ですが、


　夏休みなどなくても、仕事に生きてこそ男です。徹夜に明け暮れてください。
折り返しご返事のほど、お願い申し上げます。
　　　　　　　　　　　拝具
　　　平成 4年 7月13日
　　　　　　　　　　　　　　酒井 法雄
　様
　　　机下
　　　　　　追伸
```

図4　SAMPLE2.DOC，本文を追加

```
突然のお手紙お許しください。

　梅雨明けの暑さもひとしお、皆様にはお変わりなくご健勝のことと存じます。
私は風邪をひいてしまったのですが、次第によくなってきています。
　早速ですが、かねてからご催促の原稿は予定よりも遅れそうです。こちらも気
合いを入れて頑張っていますので、もう少しお待ち下さい。
　夏休みなどなくても、仕事に生きてこそ男です。徹夜に明け暮れてください。
折り返しご返事のほど、お願い申し上げます。
　　　　　　　　　　　拝具
　　　平成 4年 7月13日
　　　　　　　　　　　　　　　　　　　　　　　　　酒井 法雄
　編集者 様
　　　机下

　　　　　　追伸
　来月こそは早く出せるように頑張る所存です。
```

リスト17　KETUGO.DAT

```
敬具
拝具
拝白
謹白
謹言
草々
不一
拝答
敬答
謹酬
かしこ
```

リスト18　MUSUBI.DAT

```
右、用件まで。
右、ご報告まで。
とりあえずご案内まで。
略儀ながら、書中にて御礼申し上げます。
ご自愛のほどお祈り申し上げます。
お身体をくれぐれもお大事に。
皆様によろしくお伝えください。
妻もよろしくと申しております。
お返事お待ちいたしております。
折り返しご返事のほど、お願い申し上げます。
今後ともよろしくお願い申し上げます。
ご指導のほど、お願い致します。
乱文お許しください。
乱筆なもので、ワープロにて失礼いたします。
```

リスト19　TUNAGI.DAT

```
さて、
ところで、
実は、
ときに、
突然ですが、
早速ですが、
急の話ですが、
このたびは、
先日お知らせいたしましたように、
かねてからお話のございます、
先日は、
いよいよ、
私儀、
おかげさまで、
今日、
```

リスト20　WAKITUKE.DAT

```
机下
足下
みもとに
""
```


ワンラインNEWS
アドバンスド・マイクロ・デバイセズはLANの代表的規格「イーサネット」に対応した制御機能を1チップに集積した制御用ICを開発，サンプル出荷を開始した．

TEST3.BAS)を作ってみました．自動的に文章を書くとはいっても，単にひらがながランダムに書かれるだけですから，何の意味もありません．ただ，これとあらかじめ単文の入ったデータファイルとを組み合わせたら，おもしろい自動文書作成プログラムができるかもしれません．

DDEを使ったプログラミング

WordBASICでは，DDEやOLEといったWindowsの持つ魅力的な機能を使うことができます．ここでは，DDEを使ってプログラムマネージャと通信するプログラムを作ってみました．

リスト24 (PROGMAN.BAS) では，Wordが内蔵しているFileOpenダイアログボックスを使い，ファイル名を指定し，「DDE Test」というグループを新たに作成し，そのグループに指定したファイルを加えるという作業をしています．

FileOpenダイアログは，あらかじめWordに内蔵されて定義してありますから，Begin Dialogで定義する必要はありません．いきなり，「Dim dlg As FileOpen」と宣言できます．ここで指定されたファイルの名前は，「dlg.Name」で参照することができます．

DDEでほかのDDEをサポートしたプログラムと通信するときには，まず，リスト25のようにして，DDEの通信を開始します．このときの引数は，相手のアプリケーションによって異なります．こうしてオープンしたチャンネルの番号が戻り値となります．このチャンネルでこの後の作業は行われます．

DDEでの通信の実際は，DDEEx-

図5　SAMPLE3.DOC，フォントやアンダーラインで修飾

突然のお手紙お許しください。

梅雨明けの暑さもひとしお、皆様にはお変わりなくご健勝のことと存じます。私は風邪をひいてしまったのですが、次第によくなってきています。

早速ですが、かねてからご催促の原稿は予定よりも**遅れそう**です。こちらも**気合いを入れて頑張っています**ので、もう少しお待ち下さい。

夏休みなどなくても、**仕事に生きてこそ男**です。徹夜に明け暮れてください。折り返しご返事のほど、お願い申し上げます。
拝具
平成 4年 7月13日
酒井 法雄

編集者 様
机下

追伸
来月こそは早く出せるように頑張る所存です。

図6　SAMPLE4.DOC，replaceマクロで編集，SAMPLE3.DOC → SAMPLE4.DOC

突然のお手紙お許しくれ。

梅雨明けの暑さもひとしお、皆君にはお変わりなくご健勝のことと存じる。私は風邪をひいてしまったのだが、次第によくなってきている。

早速だが、かねてからご催促の原稿は予定よりも**遅れそう**だ。こちらも**気合いを入れて頑張っている**ので、もう少しお待ち欲しい。

夏休みなどなくても、**仕事に生きてこそ男**だ。徹夜に明け暮れてくれ。折り返しご返事のほど、お願い申し上げる。
拝具
平成 4年 7月13日
酒井 法雄

編集者 君
机下

追伸
来月こそは早く出せるように頑張る所存だ。

リスト21　REPLACE.DAT

```
"ください。","くれ。"
"います","いる"
"ます。","る。"
"です。","だ。"
"ですが、","だが、"
"下さい。","欲しい。"
"ください。","ほしい。"
"あります","ある"
"様","君"
```

リスト22　REPLACE.BAS，複数文字列の一括置換

```
Sub MAIN
        Dim src$(20), dst$(20)

        i = 0
        Open "replace.dat" For Input As #1
        While(Not Eof(1))
                Read #1, src$(i), dst$(i)
                i = i + 1
        Wend
        Close #1
        MAXR = i - 1
        InsertBookmark "Org"
        For i = 0 To MAXR
                EditGoTo "¥StartOfDoc"
                EditReplace src$(i), dst$(i), 1, 1, 0, 1
        Next i
        EditGoTo "Org"
End Sub
```

ecuteで行います．2つ目の引数が，相手に送るデータです．この形式も，相手のアプリケーションによって違います．詳しくは，相手のアプリケーションのマニュアルなどを参照してください．ここでは，プログラムマネージャに"[Creategroup (DDE Test)]"で「DDE Test」という名前のグループを作らせ，"[AddItem (" + f$ + ")]"で，先ほど選択したファイル名を先のグループに登録しています．

DDEでの通信の修了は，「DDETerminate」で行います．ファイルと同様に，DDEでのチャンネルも使ったらクローズしなくてはいけません．

このプログラムでは，さらにAppActivateでプログラムマネージャをアクティブにし，AppMinimizeでWordを最小化しています．

プログラムマネージャが相手では，これといった文書処理もできませんが，Excelなどを相手にすれば，データの交換などは簡単にできますから，おもしろいプログラムが作れることでしょう．

そのほかにもある多くの機能

このほかにも，WordBASICには多くの機能があります．単文登録やウィンドウ関係など，Wordならではの関数も豊富です．これらを駆使すれば，かなりおもしろいプログラムが作れることでしょう．残念ながら，今回はWordの持つ機能のすべては理解していないままWordBASICを使い始めたので，もっと突っ込んだ紹介はできませんでした．また，機会がありましたら，おもしろくてイカすプログラムを紹介したいと思います．

それにしても，WordBASICはかなり強力な処理をすることができます．しかし，動作が遅い，デバッグ環境が揃っていない，QuickBASICとは似ているがかなり独自の言語仕様と，実際に使いこなすにはなかなか辛い面もあります．また，Wordの持ついろいろな概念も十分に知る必要があります．しかし，うまく使いこなせば，必ずや能率的な文書処理をすることができるようになることでしょう．

リスト23　TEST3.BAS，自動文書作成プログラム

```
' Auto Writing
Sub MAIN
        While(i < 50)
                i = i + 1
                c$ = Chr$(Rnd() * 83 + Asc("あ") - 1)
                Insert c$
        Wend
End Sub
```

リスト24　PROGMAN.BAS，DDEを使ったプログラミング

```
Sub MAIN
        Dim dlg As FileOpen

        Dialog dlg
        f$ = dlg.Name
        chnum = DDEInitiate("Progman", "PROGMAN")
        DDEExecute chnum, "[Creategroup(DDE Test)]"
        DDEExecute chnum, "[AddItem(" + f$ + ")]"
        DDETerminate chnum
        AppActivate "プログラム マネージャ", 1
        AppMinimize
End Sub
```

リスト25

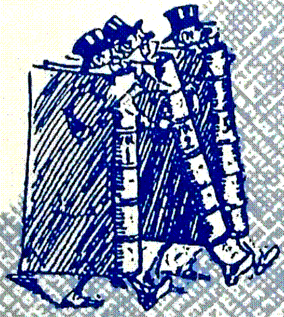
```
chnum = DDEInitiate ("Progman", "PROGMAN")
```


ワンラインNEWS
マルチメディア時代の幕開けを目前にコンピュータ業界が著作権問題への対応を迫られている．音響機器同様に機材や機器の価格に使用料を上乗せすることになる模様

IC千夜一夜

第14回

初期の国産16bitマイクロプロセッサ L-16A



虎岩 登

マイクロプロセッサは4 bitのものから8 bit, 16bitへと，順番に発展してきたように見えます．確かにその通りなのですが，16bit CPUが発表されたのは意外なほど古く，1974年にはCP1600がありました．モトローラのMC6800が発表された年だといえば，8 bit CPUとほとんど同時に開発されていたことがわかると思います．国産の16bitマイクロプロセッサでは，NECのμCOM-16やパナファコムのL-16Aが1975年に出ています．今回は，そのL-16Aを取り上げることにします．

L-16Aといっても，まず，なじみのない名称だと思います．これはパナファコムが発表したマイクロコンピュータの1つのファミリーに与えられた名称で，あるLSIを用いて構成されるコンピュータ体系に付けられた名前です．L-16Aを構成するLSIには，CPUとしてMN1610，サブチャネルアダプタ（SCA：後述）としてMN1630，DMACとしてMN1650，リアルタイムシステムコントローラ(RSC:後述)としてMN1640の，計4種類があります．これらのLSIにメモリなどを組み合わせれば，ミニコンピュータ風のシステムができあがります．

L-16Aは知らなくても，LKIT-16というマイコンキットを覚えている方がいらっしゃるかもしれません．L-16Aの評価用キットとして，1977年頃に販売されたものがLKIT-16です．98,000円の値段が付けられていたこのキットは，当時のコンピュータを使いたがっていたアマチュアないしマニアが唯一購入できる価格の16bitコンピュータであったといってもよいかもしれません．

システム・アーキテクチャ

独自の入出力チャネルを採用

L-16Aのブロック図を図1に示します．また，チップセットの写真を載せておきます．

RSCは，L-16A用のクロック・ジェネレータとリアルタイム・クロック用のタイマ，割り込み管理などを行うためのLSIです．このRSC以外のLSIは，+12V, +5V, −3Vのちょっと変則的な3電源が必要で，CPUに必要なクロックも12V近い振幅が要求されます．このクロック電圧では，RSCの出力電圧をそのまま使うわけにはいかず，別にドライブ回路が必要です．

SCAは何かといえば，まずサブチャネルというものを理解しないといけません．当時のそれなりのコンピュータ

は，CPUとメモリを接続するバス以外に，入出力機器を接続するためのチャネルと呼ばれるある種のバスを持っていました．CPUとメモリを接続するのにおもに使われるバスは，高速で同時に転送できるビット数も多いのですが，その代わりにあまり長い配線はできません．それに対して，チャネルは本来のバスに接続されるチャネルアダプタに管理される中低速の入出力専用のバスです．チャネルアダプタは，DMAの機能などを持った入出力を管理する特殊なCPUのようなものです．本来のCPUはメインメモリ内容の演算を高速で行い，メインメモリと入出力装置の間のデータ転送は（複数の）チャネルアダプタが平行して行うことで，システム全体のスループットをあげていました．

SCAは，この入出力チャネルを用いたコンピュータのアーキテクチャを，そのままマイクロコンピュータのレベルで実現しようと設計されたLSIのようです．今でいえばSCSI風の，入出力機器を接続する特殊なバスに対するインターフェースLSIです．DMACと組み合わせて，それなりに高速のデータ転送(でも200KWord/秒ですが)を行うことも可能です．

チャネルを用いると，数メートルの長さに入出力用バスを延長できますから（バッファを工夫すればそれ以上も可能)，大きな入出力機器をそれなりのデータ転送速度で接続するには便利ではありますが，CPUのバスにRS-232Cやそのほかの入出力インターフェー

写真1　L-16Aのチップセット

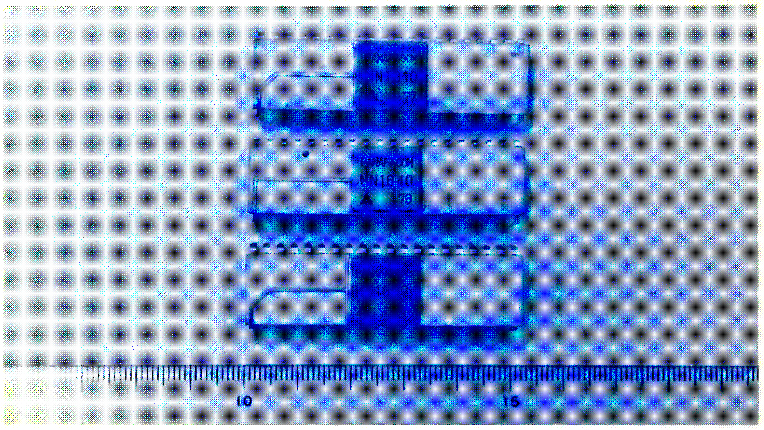
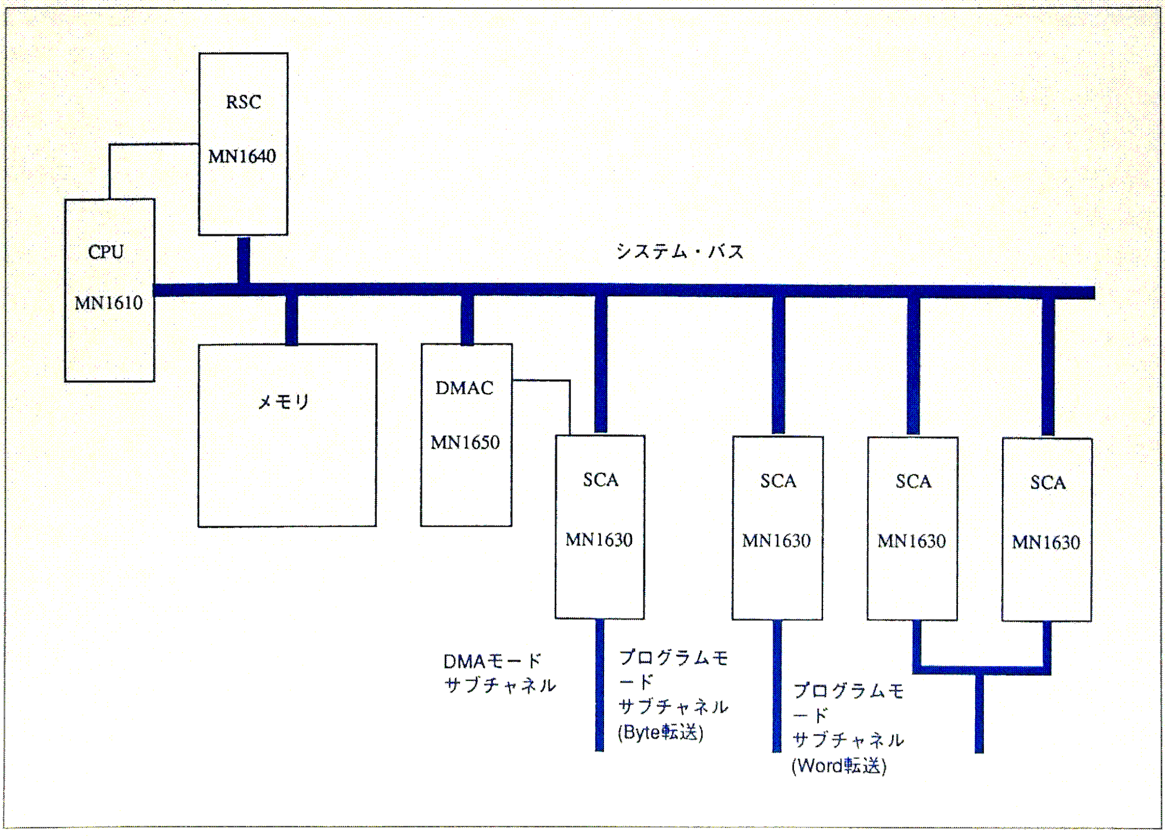


図1　L-16Aを用いた典型的なコンピュータのブロック図



ス用のLSIをつなぐほうが手軽でプログラムを作りやすいように思います．また，このチャネルは独自の規格で，GP-IBやSCSIなどのような普及している入出力バス規格ではないので，チャネルに接続する機器のインターフェースも新規開発せねばならず，その点でも効率が悪いように思います．マイクロコンピュータ時代にはちょっと時代遅れの考え方かもしれません．

CPUのハードウェア面

メモリアドレスはワード単位

CPUは写真1のとおり，40pinのDIPタイプのパッケージです．このピン数で16bitのデータと16bitのアドレスを外部とやり取りするために，データバスとアドレスバスを時分割で切り替えて使っています．データ転送は非同期型で，データの入出力サイクルを開始してから必ずメモリや周辺機器からのデータ転送が完了したことを確認する信号が来るまでウエイトするタイプです．めずらしいのは，アドレスの転送も，完了信号を必要とすることです．しかも，これらデータ転送シーケンスを開始する前にバスの使用許可を求める規則になっています．バスの調停回路はRSCに内蔵されていて，ほかのCPUやDMACとのバス利用がぶつからないように管理しています．ですから1回のバスサイクルは，

バス使用要求	(CPU)
バス使用許可	(RSC)
アドレス送出	(CPU)
アドレス受領完了	(周辺回路)
データ転送開始	(CPU)
データ転送完了	(メモリ・I/O)

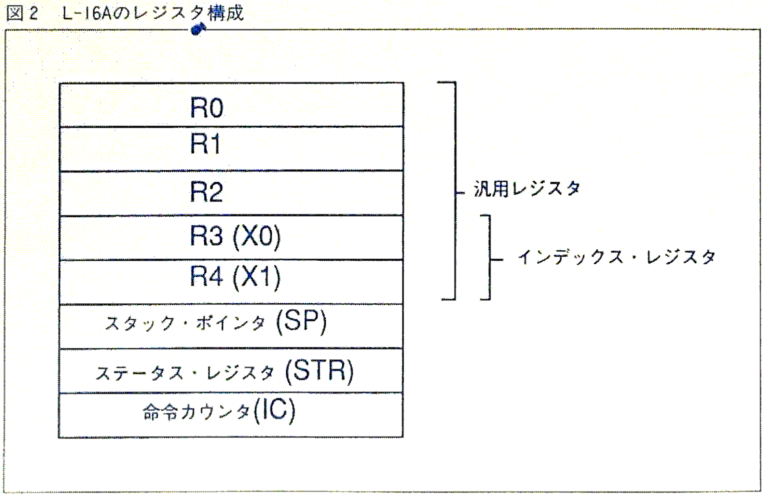
という信号をCPUと周辺が交互にやりとりするという，確実ではあるけれど手間のかかる手順が必要になります．CPUは最高2MHzのクロックで動作しますが，この手順のためかノーウエイトでのバスのデータ転送能力は500KWord/秒程度にしかなりません．

メモリはワード単位でアドレス付けされています．バイト単位でのアクセスはできません．そのため，アドレスは16bit幅で64KWord，つまり128KByteのメモリにアクセスできます．入出力のための専用命令があり，入出力空間はメモリ空間と別に256の専用アドレスが割り振られています．

割り込みは3レベル，割り込みレベルに応じた固定番地のベクタ方式によって，割り込み処理に分岐するようになっています．が，割り込みからの復帰情報も固定番地のベクタ領域に取られるので，同一レベルでの多重割り込みは困難な仕様です．

ほかにおもしろい点は，ミニコンピュータ風のコンソールパネルを実現するためか，I/Oポートから命令を読み込んで実行させるための信号が用意されています．この機能をうまく使えば，ハードウェアだけでCPU内部の任意のレジスタをパネルのランプに表示させることなどが可能になります．

図2　L-16Aのレジスタ構成



L-16Aのソフトウェア

すべての命令は1Word

ソフトウェアのほうから眺めると，もっともおもしろいのが，すべての命令が1Word命令になっていることです．半端な長さの命令はありません．

命令体系自体は，どうもNOVAというミニコンピュータの命令体系を借りてきて，ちょっとひねったような印象をうけます．ひねった分だけ使いにくくなったなんていうと，悪口になってしまいますが，少なくとも近い種類の命令体系であることは確かです．

先にレジスタ構成を図2に示します．レジスタは，汎用レジスタが5つに，特殊レジスタとしてスタックポインタとステイタスレジスタ，命令カウンタの3つがあり，合わせて8つです．汎用レジスタの内の2つはインデックスレジスタとして使用できます．

命令はすべて16bit幅に統一されていますから，そのままでは16bitアドレスのメモリを直接指定できません．そこで，0～255番地までの0ページメモリやインデックスレジスタを利用したアドレッシングが多用されます．と

くに間接アドレッシングまで備えているのは，この時代のマイクロプロセッサとしてはめずらしいと思います(が，NOVAにあった多重間接アドレッシングはありません)．アドレッシングモードを表1に示します．8種類ありますが，インデックスアドレッシングは，2つのインデックスレジスタ（X0とX1)がありますから，実質6種類です．

0ページのメモリを直接アクセスする直接アドレッシング，プログラムカウンタに8bitのオフセットを加えるプログラムカウンタ相対アドレッシング，さらにこの2種類のアドレッシングモードの間接アドレッシングモード(0ページの特定のメモリの内容をアドレスと解釈してそのアドレスのメモリを操作したり，プログラムカウンタ相対でアクセスしたメモリの内容をアドレスと解釈してそのアドレスのメモリを操作する）があります．また，インデックスアドレッシングには，8 bitのオフセットでよければ，命令の中のオフセットをそのまま利用できるモードがありますし(直接インデックス)，16bitのオフセットが必要なら，そのオフセットを0ページのメモリのどれかに格納しておいて，命令ではその0ページのメモリ番地を直接指定するモードもあります（間接インデックス)．

とにかく，上手にプログラムを組むには，0ページ領域を利用することです．16bit数値をレジスタにロードするには，イミディエート・アドレッシングがありませんから(そもそも16bit幅の命令では16bitデータを含むことができない)，その命令のそばに定数を定義しておいて，プログラムカウンタ相対アドレッシングでロードするしかありません．8 bitオフセットの範囲にないサブルーチンを呼び出すのも，プログラムカウンタ相対間接アドレッシングを使って同じようにサブルーチンアドレスを近所に定数として確保しておく必要があります．よく使うサブルーチンやシステムコールに相当するルーチンは，0ページエリアにアドレスを格納しておけば，それなりに高速にジャンプできて便利かもしれませんが，0ページエリアを食いつぶす危険があります．

ほかの特徴として，このCPUには条件分岐命令がありません．分岐命令も16bit幅ですから，アドレッシングに必要なフィールドで手一杯で，条件を記述するフィールドが命令にとれません．そこで，演算命令のほうに条件分岐を行う仕掛けがあります．演算命令の中には，スキップ・フィールドというフィールドがあります．これは，演算命令の実行結果で特定のスキップ条件が成立したら，その命令の次の命令をスキップするという動作を行うためのものです．つまり，演算命令の次に分岐命令を配置しておけば，条件分岐と同じことができるわけです．このスキップによる分岐が確実にできるのも，すべての命令が1 Wordであるという特徴のためです．

では，全命令を表2に示します．全部で33命令あります．大まかに分ければ，メモリ参照命令と演算命令，その他の命令の3種類に分けられます．非常に単純なフィールド割り当てがなされています．サブルーチン呼び出し命令がBALなんて名前になっていたり，1文字だけのニーモニックがいくつかある代わりに4文字の命令もあったりして，ちょっと違和感がありますね．

命令のフォーマットをよく見てみると，B（BRANCH）命令が命令カウンタに対するL(LOAD)命令に，BAL命令が命令カウンタに対するST (STORE)命令のようになっていたりして，ちょっとおもしろく感じます．

ハードウェア的にも16bit単位でのメモリアクセスしかできませんし，演算も16bit単位でしかできませんが，8 bitの文字コードを操作するために，BSWP命令(Byte Swap)が申しわけ程度に用意されています．

よくも悪くもミニコンピュータ風の命令体系ではあります．

表1　L-16Aのアドレッシングモード

アドレッシング・モード	意味	有効番地
0	0ページ・ダイレクト	D
1	命令カウンタ相対	(IC) + d
2	0ページ間接	[D]
3	命令カウンタ相対間接	[(IC) + d]
4	インデックス	(X0) + D
5		(X1) + D
6	間接インデックス	(X0) + [D]
7		(X1) + [D]

dは8bitの符号付き整数(-128~127)を表す。
Dは8bitの符号無し整数(0~255)を表す。

国産初の16bitパソコンに使用

開発環境の充実が鍵に

L-16Aは，1978年に作られた国産では16bit初の？　パーソナルコンピュータC-15（パナファコム社製）に使わ

れています．一般用というより，計測・制御・科学技術計算用に作られたもので，GP-IBを備えて計測機器と接続可能にして，補助記憶装置にデジタルカセットテープを内蔵し，さらにそれぞれの目的別のBASICをカセットテープで供給するという，たいへんに特徴のあるコンピュータでした．たとえば科学技術計算用のBASICにはMAT文があり，行列演算が可能でしたし，計算用ライブラリも供給されていました．まぁ，そんなわけで，あまり一般の方には目に触れることもないコンピュータであることは確かです．

その後，パナファコム社製のビジネスコンピュータにも使われています．また，MN1610のメモリ空間を256KWordにまで広げて高速化したMN1613とか，浮動小数点演算機構をCPUの内部に組み込んでさらに高速化したMN1617のような改良品もありました．MN1617などすばらしい性能ではありましたが，いかんせんマイナーなCPUのためか，ほとんど人に知られずに消えていっています．なお，これらの改良品もそれぞれパナファコム社製のビジネスコンピュータに使われて販売されていました．もしかすると，富士通や松下の名前のパーソナルコンピュータとしてオフィスでそれと知らずに使っていた方もいらっしゃるかもしれません．

あるCPUが市場を支配するには，そのCPUをサポートするソフトウェア開発環境が，ハード・ソフトともに多くのメーカーから出荷されて，競争を行う状況にならなければいけないのですが，やはり1社だけの力ではどうしようもなかったということでしょうか．日本はCPUの開発能力が低いということになっていますが，実際はCPUというLSIの開発設計能力は十分にあると思います．ただし，CPUという製品は，コンパイラやOSなどソフトウェアの開発環境が大量に供給されなければ実用にはなりません．すでに，それなりの能力を持った開発用ソフトウェア・メーカーは，現在主流になっている少数のCPU用の製品を開発するのに手一杯で，よほど新しいCPUが主流になりうる保証がなければ新規に製品を開発してくれません．なにしろ，現在主流のCPU向けの開発環境は，ほかの会社と熾烈な競争状態にあるわけで，手を抜けないのですから．

表2　L-16Aの命令表

ニーモニック	命令フォーマット	動作内容	
L	[11][M][R][D/d]	メモリからレジスタへロード	
ST	[10][M][R][D/d]	レジスタをメモリへストア	
B	[11][M][111][D/d]	分岐	
BAL	[10][M][111][D/d]	サブルーチン呼びだし	
IMS	[11][M][110][D/d]	メモリをインクリメント	結果が0ならスキップ
DMS	[10][M][110][D/d]	メモリをデクリメント	
A	[01011][Rd][Skip][1][Rs]	加算	
S	[01011][Rd][Skip][0][Rs]	減算	
C	[01010][Rd][Skip][1][Rs]	比較	
CB	[01010][Rd][Skip][0][Rs]	下位バイトどうしの比較	
MV	[01111][Rd][Skip][1][Rs]	レジスターレジスタ間転送	
MVB	[01111][Rd][Skip][0][Rs]	下位バイトだけのレジスタ間転送	
BSWP	[01110][Rd][Skip][1][Rs]	上位・下位バイトを交換して転送	
DSWP	[01110][Rd][Skip][0][Rs]	bit4-7とbit8-11を交換して転送	
LAD	[01101][Rd][Skip][1][Rs]	BCD演算の補正用命令	
AND	[01101][Rd][Skip][0][Rs]	論理積	
OR	[01100][Rd][Skip][1][Rs]	論理和	
EOR	[01100][Rd][Skip][0][Rs]	排他的論理和	
SL	[00100][R][Skip][1 1][EM]	左シフト	
SR	[00100][R][Skip][1 0][EM]	右シフト	
SBIT	[00111][R][Skip][BP]	ビット・セット	
RBIT	[00110][R][Skip][BP]	ビット・リセット	
TBIT	[00101][R][Skip][BP]	特定ビットのテスト	
AI	[01001][R][Skip][Imm]	定数加算	
SI	[01000][R][Skip][Imm]	定数減算	
LPSW	[00100 000 000000][LV]	割り込み復帰	
H	[00100 000 00000000]	CPU停止	
PUSH	[00100][R][00000001]	レジスタのプッシュ	
POP	[00100][R][00000010]	レジスタのポップ	
RET	[00100][R][00000011]	サブルーチン復帰	
RD	[00011][R][I/O addr]	I/Oの読みだし	
WT	[00010][R][I/O addr]	I/Oへの書き込み	
MVI	[00001][R][IMM]	バイト定数のレジスタロード	

というわけで，特殊な組み込み目的ならいざしらず，汎用コンピュータ向けCPUの開発は，売り方が一番難しいのです．

最後に，ユニークな1つの国産CPUに，黙禱．

Book Review

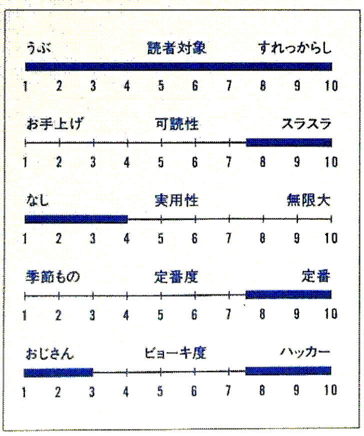
■中村正三郎

信じられないエピソードの連続で半導体王国・日本の歴史を再現

**■電子立国
日本の自叙伝
(上・中・下・完結)
各1,500円
相田洋著
日本放送出版協会**



■評価チャート



話題作なのですでにお読みになった方も多いと思うが，このたびようやく完結巻が出版されたので，ここで半ばご祝儀として紹介する．この大変な労作に少しは報いたいと思ったからである．もしまだお読みでない方は，ぜひお読みください．本書の面白さは，多くの人の折紙付きである．

この本のルーツは，NHKで放映された同名のスペシャル番組「電子立国日本の自叙伝」のシリーズにある．現在，世界最高水準の半導体王国になった日本．その黎明期はどうであったのか．トランジスタの発明，実用化に日本がどう取り組んだか，ICの時代はどうだったか，さらにLSIの時代になってどうなったのか，電卓戦争を経てマイコンの時代まで，番組は時代の現場にいた人々，特に研究者，技術者の貴重な証言を交えて構成されていた．この番組を制作したのが，本書の著者である相田洋氏である．時の流れとともに少なくなっていく当時の証言者，製品，試作品の実物を求めて世界中を飛び回り，インタビューを敢行し，写真を撮り，再現ドラマを作り，半導体の歴史，半導体に関わった日本の歴史を立体的に再現することに成功した．

テレビ放映を観たとき，今からでは信じられないエピソードの連続に圧倒されたものだが，評者は本書によってあの感動を再び味わうことができた．

本書は，テレビの単純な活字版ではない．全体の流れはテレビのものだが，その裏側にあったエピソード，制作時の苦労が生き生きと描かれ興味は尽きない．

ビデオに取ってじっくり観たつもりだったが，やはり見落していたこと，忘れていることも多々あった．活字による再現もまた有用だとの思いを新たにした．

我々，計算機屋，中でもソフト屋に関係してくるのは，完結巻である．若き日の嶋氏なども登場するインテルのマイコンの話からである．このあたりからは，いろんなエピソードをご存じの方も多いだろう．がしかし，上巻のゲルマニウムトランジスタ，中巻のシリコントランジスタ，下巻のIC，LSIと電卓戦争などなど，それ以前の話もすさまじい．きっと虜になるはずである．

当時の関係者は，今だから笑って話せるといった面持ちで証言しているが，評者が当時の現場にいたら，髪は抜け，目はつぶれ，胃に穴はあき，血尿は出るわ，尻から出血するわの事態になること必定の物語が満載である．

技術者の文字どおり，血と汗と涙の結晶が半導体になっていくのである．

NHKに問い合わせたところ，「電子立国日本の自叙伝」のビデオはまだなく，予定も未定とのことである．したがって見逃した方は，本書を手に取るしかない．先人の苦労を知り，気持ちを新たにするのはよいことだと思う．評者は，本書からそれまでの常識から脱却することが，世界最高水準の頭脳であっても，どれほど大変なことか，教わった．技術者ならずとも一読をお薦めする．

ワンラインNEWS
富士通はパソコン通信ソフトを内蔵し，通信機能を強化ポケットサイズのワープロ「OASYS Pocket 2」を発売．470グラムと従来よりさらに10%以上軽量化した．

Windows 3.1, QuickTimeの登場で注目されるマルチメディア関連図書

古田島義和（神田・書泉グランデ）

Windows 3.1の登場で，MS-DOSパソコンの世界でもマルチメディアが可能になりつつあります．一般ユーザーにもインパクトのある利用法として，マルチメディアをパソコン普及の目玉にしていこうというのがメーカーの戦略のようです．一方，Macintoshは，QuickTimeというマルチメディアのためのソフトウェア・プラットフォームで，マルチメディアの世界で圧倒的にリードしている現状を，さらに発展させようとしています．処理能力では本命と見られているUNIXマシンも，すぐれたGUI環境をもつX-Windowのもとでマルチメディアは最も力を入れているテクノロジーのひとつです．今後のコンピュータ環境にとって，マルチメディアが重要なキーテクノロジーのひとつです．しかし，ここ2～3年，大きな注目を集めていますが，「何ができるのか」といったあたりが，いまひとつはっきりしないことも事実です．マルチメディアといった時に，思い描くものが人それぞれバラバラなのではないでしょうか．CD-ROM，光磁気ディスク，ビデオ機器，FM音源，MIDI楽器などととの接続が可能になり，従来は磁気ディスクに限られていた記憶メディアが大容量な各種メディアにまで広がってきました．また，各種の映像や音響関係機器との接続が可能になってきました．

また，ソフトウェアにより，ビデオ映像，グラフィックス，アニメーション，音声，音楽，音などを組み合わせ，制御が可能になり，いままで文字と数値，グラフ，単純なグラフィックスなどに限られていた表現手段から，より多彩で複雑な表現が可能になりました．さらに表示，表現，検索の仕方や見え方が階層構造化され，たがいにリンクされて，従来のメディアでは不可能であった構造，表現法の実現がこれからは必要不可欠になっていくでしょう．マルチメディアに関する図書は次の通りです．

MicrosoftPress社の「Interactive Multimedia」，「Learning with Interactive Multimedeia」，Que社の「Que's Macintosh Multimedeia Handbook」，Academic Press社の「Hypertext & Hypermedia」，新刊では，Addison Wesley社（以下AW）の「Interactive Multimedia Computing」（Blattner著）や，今年秋以降に発刊が予定されている同社のPhilips CD-I Series（3分冊），「Introducing CD-I」，「The CD-I Production Handbook」，「The CD-I Design Handbook」などがあげられます．今後マルチメディアの広がりが期待されます．

Macintoshの新シリーズを紹介しましょう．AW社の「Inside Macintosh Series」が「New Inside Macintosh (Apple Technical Library)」として，92年11月ころから93年はじめにかけて6点出版されます．「Inside Macintosh:OverView」，「同：Toolbox Essentials」「同：Files」，「同：Processes」，「同：Memory」，「同：Imaging」で，このほかに予定されているものには「同：QuickTime」，「同：Operating System Utilities」，「同：Text」，「同：Additional Features of the Macintosh Toolbox」，「同：Interapplication Communications」，「同：Networking and Communications」，「同：Devices」があります．ほかに同社から「The Macintosh Dictionary」が近日入荷の予定です．また，米国でも日本でもともにベストセラーになっているのが，IDG出版の「INFO World, Windows 3.1 Secrets 2/E」で，Windows 3.1がユーザーにどのくらい期待されるか，注目していきたいと思います．

■和書&洋書ベスト10　　（書泉グランデ調べ：1992年5月1日～5月31日）

	和書	出版社		洋書	出版社
1	UNIXネットワークベストプログラミング入門	技術評論社	1	New Hacker's Dictionary	M.I.T.
2	BERKELEY HACKERS	ピレッジセンター出版局	2	Object-Oriented Modeling and Design	Prentice Hall
3	SEクライシス	オーム社	3	C++ Programming Language 2/E	Addison Wesley
4	GNU Emacs Lisp Manual	ピレッジセンター出版局	4	Zen & The Art of Resource Editing	Peachpit Press
5	NIFTY-Serveデータライブラリ・ディレクトリ'92年前期版	星雲社	5	Foundations of Computer Science	W.H.Freeman
6	アラン・ケイ	アスキー	6	Modern Operating Systems	Prentice Hall
7	カオス―新しい科学をつくる	アスキー	7	UNIX Network Programming	Prentice Hall
8	C言語によるはじめてのアルゴリズム入門	技術評論社	8	TOG on Interface	Addison Wesley
9	IBM PC/AT互換機ガイドブック	ソフトバンク	9	Windows 3.0 Secrets	IDG Books
10	オブジェクト指向狂詩曲	技術評論社	10	Programming Perl	O'Reilly

ワンラインNEWS
コンパックはCPUに米インテル社の「386 SL」を使い 40 MBのHDDを内蔵したノート型パソコン「コンテュラ」など4機種8モデルを発売した．

★★★佐々木哲の★★★ U.S. プロダクツウォッチ

★WPが通信ソフト会社を買収

ここのところ，毎月のように会社買収のニュースが伝えられます．今月はワードパーフェクト社がFAXモデムソフト，ソフトウェアMNPエミュレーション付き通信ソフトなどで知られるMagicsoft社を買収したというニュースです．

同社は，以前この欄でもお伝えしたMTEZやExpress-FAXなど，技術的に高度でかつ安価な通信ソフト群を開発・販売してきました．通信ソフトMTEZは，コンパクトながら無数のターミナルエミュレーション，ファイル転送プロトコルを持ち，またソフトウェアでのMNP 5のエミュレーションをも実現していました．また，FAXソフトであるExpressFAXオプションは，インテルのFAX APIであるCASをclass 2モデムなどに対しても実現するなど，異色を放っていました．これらのソフトは単体販売だけではなく，Dell，Megahertz，Zoomなど多くのモデム，FAXモデムなどにバンドルされています．

これによりワードパーフェクト社は，一気に通信分野へも進むわけです．ワードプロセッサだけでなく，統合ソフトをDOS，Windows，Unixなどに持つ同社にとって，これは強みですね．すでに3月から出荷されている統合ソフト，WordPerfect Worksでは昨年から共同作業していたそうです．

このところ，ロータスと同じく単一商品病におかされているかのような印象を受けるワードパーフェクト，WordPerfect for Windowsも好調に100万本目の出荷を終え，各社との連携も深めるとの動向も伝わってきます．頑張って欲しいものです．

★イメージ入力に統一規格，TWAIN

ワードパーフェクト社などの支援を受け，今まで無法状態だったイメージ入力にTWAINという規格ができつつあります．今ではフラットベッドなど各種のスキャナが市場にあふれていますが，どれも専用のドライバを必要とするものでした．この状況を打開するために1990年頃からAldus，Caere，Eastman Kodak，HP，Logitechが中心になり，ワークグループが結成され活動してきました．目的は，どのアプリケーションからでもイメージデータが読み込める統一APIを規定することです．

対象プラットフォームは，まずはMacintosh，続いてWindows，さらにはUnix，OS/2などを目指しています．また，ハードウェアもハンドスキャナからフラットベッドスキャナ，フレームグラバーまで広い範囲のものが対象です．今では175以上のハードウェア，ソフトウェア会社が参加しているそうです．この6月末のPC ExpoでもインテルはそのFAXソフトウェアでのサポート，またMicrografxなどもTWAINサポート製品の出荷をアナウンスしています．このTWAINをサポートしたソフトウェアであれば，同じくTWAIN APIを持ったどのスキャナでも使えることになります．

このAPIについての資料は，CompuserveではHP Peripheral forum（GO HPPER）で入手することができます．現在，仕様書のほかにMacintosh，Windows用のツールキットが提供されています．

★やっぱりお勧め，インテル社のFAXモデム

以前からこの欄でご紹介してきたように，V.32bis/V.17（14400bps）のFAX付きモデムが超格安で出回り始めました．Supra，Zoomなどの名前を一部秋葉原でさえ見かけるようになっています．

ハードウェアは確かに日本のメーカー製のように注意深くは作られていませんし，実際に熱を持ったり繋がらないホストがあるなどの障害も多く聞かれます．

今の異常ともいえる“安ければいい”というDOS/V互換機の氾濫にも通じることですが，品質と価格のバランスを巡ってはしばらくは混乱が続くでしょう．それでもいままでこの競走原理が効かなかったとさえいえる日本のモデム事情，少しはまともになることを祈ります．

もっとも，これらの高速格安FAXモデム群，ソフトウェア的になんら新しいものがあるわけでもありませ

ワンラインNEWS
日本電気はカラーラップトップ型を含むWSとサーバ合計5機種と，それらに搭載するOSを発売．このうち上位機にはRISC型CPU「R4000」を搭載する．

ん．今までの通信ソフトに，Windows/DOS用のFAXソフトウェアを使うだけ．DelrinaのWinFAXなどですね．とくに，FAXの安定した受信はいくら高速のマシン上でWindowsを走らせていたとしても負荷が大きいものです．

FAXを安定して受信したりするには，INTEL SatisFAXtionのようなCPU付きのインテリジェントなものにはかないません．表のアプリケーションの実行には全く影響を与えず安定してバックグラウンドで送受信できます．

インテルのSatisFAXtionの製品ライン[*1]も大きくモデルチェンジしています．ここで整理しておきます．

この4月にインテルのFAXモデムラインは4つのモデルになりました．上位からSatisFAXtion Modem/400，/200，/100そして外部式の/400eです．/200モデルは従来のSatisFAXtionと同等のもので，オンボードに80186CPUの載ったATバスカードです．モデム部分は，V.42bis付きの2400bps，FAXは9600bpsです．添付ソフトウェアが一新し，エプソンだけではなく，HP PCL5のプリンタエミュレーションが付き，また，Windows用のFAXソフトなども新たにインテル製のものが用意されています．

/400は，V.32bisを持った最上位機種です．DTEスピードは115Kbps以上までサポート，またSmart UARTと称し，インテリジェントなバッファ付きUARTとしてPC側に見えるようになっています．そのためWindowsなどのバックグラウンドでも安定して高速な通信ができます．FAX部分はV.17だけでなく，FAX モデムとしてはめずらしくMR圧縮までサポート．より高速な送受信ができます．

/100モデルは，今までのSatisFAXtionとは違う，安価なCPUなしのFAXモデムです．FAX部分も受信は4800bpsまでと割り切っています．同じインテルのFAXソフトウェアのほかに，おまけとしてCrosstalkの通信ソフトが付いてきます．

/400eモデルは，インテルFAXモデムでは初の外部式のものです．とはいえ，CPUは載っていません．V.32bisとV.17の14,400bpsまでのFAXが可能です．

/400と/200にはCPUが載り，安定したバックグラウンドでの送受信をサポート，また添付ソフトウェアとオンボードのLogitec互換スキャナポートにより，上に述べたTWAIN対応のイメージ読み込みもサポートしてます．またFAX，モデム，電話の切り替え機能も持っています．

/400eは，通常の外部式FAX モデムですが，Flash ROMにファームウェアを載せ，ROM差し換えなしでアップグレード可能にするなど特徴を出しています．

Windows，DOS用に全く新たにFAXソフトウェアが用意され，どのモデルにも付属しているのがいいですね．もうハードウェアだけで競争できる状態ではありせん．あと面白い点としてはFAXインターフェースが今までのCASに加え，どれもEIA class 1のサポートとなりました．うーん，やっぱり最上位機の/400はすごそうです．

*1 : SatisFAXtion Modem/400e	$549
SatisFAXtion Modem/400	$499
SatisFAXtion Modem/200	$369
SatisFAXtion Modem/200 MicroChannel	$549
SatisFAXtion Modem/100	$129

Intel Corporation
PC Enhancements
CO3-07
5200 N.E. Elam Young Parkway
Hillsboro, OR 97124-6497
phone (800) 538-3373

★もっとFAXモデム!!

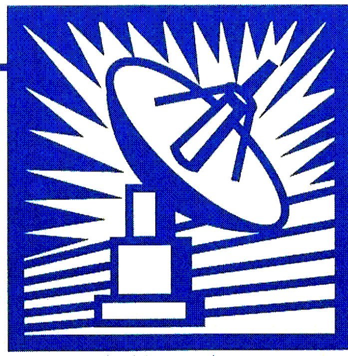
これだけ広まってくると，安価なV.32bis，FAXモデムの話だけでは面白くありません．少し毛色が変わったものをご紹介しましょう．バッテリ駆動で持ち運び可能なポータブルのものです．

サンタクララのMicro Integrated Communications[*2]は，重さ6オンス（230g）のV.32bis，V.17，V42bisのFAXモデムを799ドルで発表しています．FAX部分はEIA class 2対応です．同社はv.32のモデムだけのバージョンも349ドルで発売しています．同様の製品は，Macronix，Megahertzなどからもでています．

USのFAX モデムは，9Vバッテリで動作し，せいぜい3時間くらいしかもたないものがほとんどです．オムロンがOmron Impala 24/96[*3]という名前で2400/9600bpsのポータブルFAXモデムを発表しています．重さは150gほどで4本の単3電池で動くところがみそです．バンドルされているソフトウェアは，DelrinaのWinFax Lite，DosFax Lite，そしてBit SoftwareのBitCom for DOSです．

*2 : Micro Integrated Communications
phone (408) 980-9565
fax (408) 980-9568
*3 : Omron Impala 24/96 $399
Omron Office Automation Products Inc
phone (408) 727-1444

Illustration : NOBUKI FUKUHARA



西川和久の
希望的観測

特選!発表会2つ。オリベッティのRISCマシンとエプソンからもPC互換機が

1 R4000搭載のACEマシン 日本語Windows NTが待ちどおしい!!

日本オリベッティは6月4日に「RISC PC M700-10」の発表会を行った。50MHzのR4000を搭載しACE仕様なのは当然であるが，デベロッパーズ向けのシステム内容，価格設定の凄いこと!!　RAM 16MB/HDD 210 MB/CD-ROM Drive/3.5インチFD 2.88MB/SCSI2/Ethernet/ステレオサウンドシステム/1280×1024 256色ノーインタレスのグラフィックシステム・17インチモニタ/EISAバス×4/Keyboard/Mouse……これだけのものがすべて揃ってたった91万円!!　これは，現在筆者が使っている486DX/33MHzのシステムとCPU以外はほぼ等価であり，値段も大差ない。もちろんマシンパワーはケタ違いである(^^;　とくに1280×1024 256色ノーインタレスのグラフィックシステムと17インチのモニタが込みだというのは驚異である。このグラフィックシステムは最近PCの世界で流行しているアクセラレータ・タイプではなく，RISCチップのデータバスに直結している64ビットのメモリ空間へビデオRAMがマッピングされている。ちょうど，32ビットローカルバスと考え方が同じであるが，データ転送レートはその比ではない。一言「お金があったら欲しい!!」マシンなのだ。

しかし，不安要素もある。潰しがきかないことである。今，対抗可能なマシンとしては，年中に出るであろう586マシーンであれば別にNTやOS/2など，32ビットOSで使わなくとも最悪，超高速DOSマシンとして使えるのだ。今後のOS戦争など，どうなるか誰もわからないが，少なくとも当面DOSは残るハズである。安価なRISCマシンとハイエンドPC互換機の唯一の違いはこの点なのだ。

当日，Windows NTのベータ版などをデモしていたが，十分なパフォーマンスであり，ワークステーションとして使えばかなり快適な環境であること間違いなしだ!!　高級マルチメディアマシンとしても十分遊べるであろう。結局この「遊べる」が筆者にとっては（ほかの方はこんな不純なことを考えないと思うが……　^^;）重要である。いかに高速なマシンが手に入ろうと，その上で使うソフトウェアのジャンルは仕事の関係上おおよそ決まっている。ワープロ，表計算，通信，開発環境，またマルチメディア関連といってもMUSIC CDをコンピュータからコントロールしたり，システムイベントによって音を鳴らしたり，モニタ上でテレビを見たり……この程度の使い方であり，現在の環境でも十分といえば十分なのだ。最近，DOS/Vが出たおかげでエンドユーザーまでもが486マシンを気軽に使うようになってきたため「ならば次は!?」と考えているだけなのだ。ただし，Unix系は個人的に好みではないために，現在OS/2かNTしか選択の余地がなく，RISCマシンやEISAバスマシンなどにその興味の対象が向いているだけである。

ワンラインNEWS
日本アイ・ビー・エムは情報システムのソフト開発などSEサービスを提供する100%出資の子会社「日本アイ・ビー・エム東海システムズエンジニアリング」を設立した。

2 これで先は見えた! 決戦は1995年だ!

PC-98陣営最後の砦，EPSONもついにPC互換機を扱うことになった．もちろん，OADGにも参加する :-)．第一段は日本DECからのOEMであるが，米国自社ブランドPCを国内へ持ち込むのも時間の問題だと思われる．大儀名文は富士通と同じく「ネットワーク用として」となっているが，その富士通でさえもビジネスショウにおいては，ICLのサーバーマシンを使って日本語Windowsを動かす始末……もうこまでくれば，NECを除く(?)各メーカーすべてのラインナップがPCアーキテクチャになるまで3年とかからないであろう．某ネットでは一時期「PC-98とPC互換期のシェアが何時逆転するか!?」の議論があったが，今でも著者の意見は変わらない．やはり決戦は1995年である．

ある雑誌のアンケートによると「486マシンを買い替える時に何を選ぶか？」では，PC-98は全体の約30%しか候補に上がっていない．面白いことにその横にあった「今後導入予定のOS」はMS-DOSが30%程度……ピッタリ数字があってしまうのだ．もちろん，そのほかはPCアーキテクチャマシンであることはいうまでもない．こうなると何時頃486マシンが普及するかを考えれば，おのずと答は決まってくる．まあ，来年のこの時期には486DX/33マシンなんて，パワーユーザーの間ではゴミマシン扱いされているに違いない．(^^;

何度もいっているが，PCアーキテクチャに切り換えることにより，Windows，PENコンピュータ，Windows NTやOS/2 2.0に代表される32ビットOS，NetWare，LAN Managerなどのネットワーク，そしてパームトップ……今後パソコンの主軸となる環境のほとんどが米国版であればそのまま動き，日本語化も安易でかつ安定作動する．これまで10年間，日本だけが全世界から孤立して独自路線で過ごせたのは，DOS 3.xが主だったからであり，そのDOSでさえ5.0以降は危なそうな雰囲気である．

最近DOS/V叩きも多くみられ「それでもPC98!!」という記事もよく見かけるが，これにはただし書きを付けていただきたい．“スタンドアロンでかつDOS上の日本語処理をするには最も適したパソコン”と．筆者はこの一文があれば，何も文句をいうつもりはない．さらにもう一言，DOS/VのCONFIG.SYSを見て「最小構成の必要行数が複雑である．故にDOS/Vは難しい」とよく書かれているが，これも間違い．確かに日本語を表示するために，$FONT.SYS，$DISP.SYSの2つは追加されているものの（かな漢字用の$IAS.SYSやANSI.SYSは必修でない），COUNTRY＝やKEYB.COM/KEYBOARD.SYSなどは，本来DOS4.x以降のスペックであり，SBCS圏のキャラクタやキー配置をマルチリンガル対応するために追加された機能なのだ．このような記述がCONFIG.SYSから消えているのは，単にスペックダウンしているだけである．このため，DOS/V固有のドライバを除けば全世界的に使われているCONFIG.SYSの内容なのだから，日本人にだけ特別難しいわけがない．

いずれにしても，ここ1年半ほどいろいろな雑誌でDOS/V関係を大騒ぎしてきたが，現時点で完全に先が見えたようである．業界の方と話をしていて，筆者が「1995年には……」などいうと，少し前までは「そんな事絶対にない!!」という意見が大半であったが，最近人によっては「そんなにかかりますか!?」と答が返ってくるほどなのだから…… :-)．



Computer Laws

コンピュータの法律

コンピュータ取引の法律問題

◆

熊谷信太郎(弁護士)

■ソフト使用許諾契約の問題点

ソフトウェア関係の契約は，ソフト開発契約，ソフト使用許諾契約，ソフト譲渡契約に大別することができますが，ここではソフト使用許諾契約の問題点について説明したいと思います．

ソフト使用許諾契約は，ソフト所有者が有償でソフトの非独占的使用権をユーザーに許諾するものですが，その法的性質については議論があるところです．

契約の法的性質については，民法という法律が原則を定めていますが，民法は，明治時代に作られた法律ですから，コンピュータ関係の契約のことを念頭において制定されているわけではありません．したがって，民法が定めている13種類の典型的な契約にソフト使用許諾契約がピッタリ当てはまるというわけではありません．

その中で，あえて近い契約の類型を探すとすれば賃貸借契約ですが，ソフト使用許諾契約と賃貸借契約はその本質的部分が異なります．賃貸借契約は，土地や家などの貸し借りの際に用いられるポピュラーな契約ですが，そこでは，借主が賃貸借の目的物を独占的に占有して，支配することが予定されています．たとえば，マンションを借りた場合，借主にマンションを独占的に占有して使う権限があり，貸主である大家は，マンションの所有権は持っていますが，占有する権限はありませんし，ほかの人に同じマンションを貸して賃料を二重にもらうということもできません．

ところが，ソフト使用許諾契約の場合，借主はソフトを使うことができますが，自分だけで独占的に使用することができるわけではなく，オリジナルのソフトを持っている貸主が，ほかの人に使用許諾した場合には，ほかの人も同様の使用権をもつことになります．この点が，ソフト使用許諾契約と賃貸借の大きな相違点といえるでしょう．

もっとも，これは典型的な例をとって説明していますが，ソフト使用許諾契約の内容は法律で規制されているわけではありませんから，契約で独占的な使用許諾を行うことは可能です．その場合には，契約の内容は賃貸借に接近することになります．

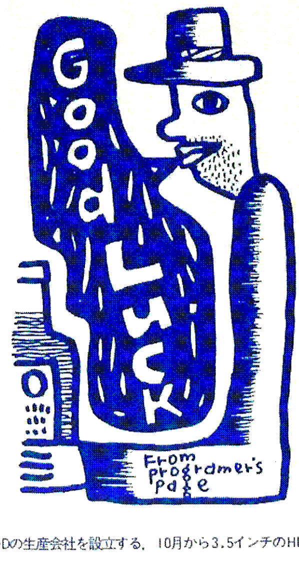
■代理店方式と直接契約方式

具体的なソフト使用許諾契約の方式には，大別して，代理店方式と直接契約方式があります．

代理店方式の場合，メーカーと代理店がソフトの使用許諾契約を締結し，代理店はさらにそれをユーザーに再度使用許諾するという方式をとる例と，メーカーがプログラムの複製物を製作して，それを代理店に売り，代理店がユーザーに転売するという方式とがあります．後者は，使用許諾という形式からはずれて，売買という形をとるわけです．

これに対して，最近パソコンのプログラムなどでは，直接契約方式が非常に多くなっています．

直接契約方式とは，メーカーからユーザーに直接使用許諾する方式で，具体的には，プログラムの複製物をビニ



Illustration：Nobuki Fukuhara

ワンラインNEWS
松下寿電子工業は，アイルランドに小型コンピュータ用のHDDの生産会社を設立する．10月から3.5インチのHDDの生産をはじめる予定．

ール袋などのパッケージに入れて販売する際に，パッケージを破らなくても契約条項や注意書きが外から読めるようにしておき，同時に，契約条項の中に「パッケージを開封すると，契約条項に同意したものとみなし，使用許諾契約が成立する」という条項をいれておくという方法で行われており，通常，「シュリンク・パック契約」あるいは「シュリンク・ラップ契約」といわれています．

シュリンク・パック契約においては，パックを破ることが使用許諾契約の承諾になるのかという議論がありましたが，現在ではパックを破った時点で承諾が成立し，契約条項はユーザーを拘束するという方向で見解がまとまりつつあります．

シュリンク・パック契約の特徴は，契約当事者がメーカーとユーザーであり，代理店は契約当事者として登場してこないという点にあります．代理店が中間に介在しないため，使用許諾した数や使用許諾料金をメーカーが的確かつ用意に把握することができ，品質の保証とバージョンアップなどのサービスの確実な履行が可能になるというメリットがあります．

アキバ短信

営業時間・定休日は店により異なる
(ただしほとんどは年中無休)

秋葉原ラジオ会館
アキバのマイコンショップ発祥の地の「現在」

今月は残念なニュースをいくつかお伝えする．まず，先月アキバ短信特別版で報告した，香港486マシンであるが，ちょうど先月号が刷り上がったその日に虎岩氏から連絡があって，現在はまったく立ち上がらない状態であるという．当初はハードディスクのトラブルだろうくらいにたかをくくっていたが，どうやらROMそのものに異常が発生している模様だとのこと．保証書には「店またはサービスセンターへの持ち込み修理のみ」と明記されている．故障したマシンを香港までもう一度持っていくか，CPUをのぞいたマザーボード全部を取り替えるよりほかはない．少なくとも虎岩氏にとって香港486マシンはそう安い買い物とはならなかったわけだ．

2つめのニュースは，あるパソコンショップの閉店である．秋葉原ラジオ会館5階の「マイコンpulse」は，ソフトウェアが定価の25%引きで，中でもコンパイラ製品の割引率がよく，ハードディスクやモデム，メモリボードなどの品ぞろえも比較的豊富な，いわば中堅どころのパソコンショップであったが，6月はじめに突然閉店してしまった．閉店の原因は，店を維持していくだけの売り上げがなくなったからということらしい．

ラジオ会館（通称ラジカン）は，秋葉原パソコン（マイコン）ショップ発祥の地といわれ，老舗のパソコンショップが多い．JRの駅の目の前という立地条件，7階にはNECのPC-98シリーズの新製品を展示するBit-INN東京があることなどから，ハード，ソフト，周辺機器から新製品情報までが揃うパソコンユーザー必見スポットであったが，最近では，1店でハード，ソフト，周辺機器から書籍までを揃えたラオックスTHE COMPUTER館にその地位を譲り渡してしまった感がある．実際のところ，ここ数年の間に閉店した店も多い．中には逆に改装して店舗を広げたり，2店めの店舗を開いたところもあるが，ラジカンを訪れる人の数は一時期よりもかなり減っていることは感じられる．時代の流れとはいえ，かつての賑やかさを知る者にはさびしい限りだ．

3つめは，本連載が今月号をもって終了することである．プログラマ御用達の店を紹介するという当初の目的から逸脱することも多々あったが，この1年おつきあいくださった読者の皆様には感謝いたします．

最後にひとつ，お知らせを．小社ではこの8月に『Programmer's Page』の別冊として秋葉原のパソコンショップを徹底取材した「秋葉原電脳市場」という本を上梓することとなった．アキバの街を探検するためのガイドとして活用していただければ幸いである．



●プログラマの仕事場 17

トムキャットコンピュータ株式会社

AT互換機で98ソフトを動かす Virtual-98で注目を集める

先ごろトムキャットコンピュータから発表されたVirtual-98が注目されている．ATバス用のボードとソフトウェアによって，PC/ATまたはその互換機でPC-9801を仮想マシンとして実現するこの製品は，日本のパソコンマーケットにどのような影響を与えるのだろうか．今回は，同社を訪ねて，製品の詳細と今後の展開についてうかがった．

マイコンキットからはじめて互換技術の開発へ

1987年にパソ協から発表されたPC/AT，PC-98両互換機，1990年にASTリサーチから発売されたPC/AT, PC-98両互換機……．これらの互換機には，いずれもトムキャットコンピュータが関連していた．というより，同社が開発した互換技術によって，こうした両互換機が成り立っていたのである．製品発表の折りに名前が登場したトムキャットコンピュータとは，どういう会社なのだろうか．そう考えていた読者も少なくないだろう．

同社の創立は，1976年にまでさかのぼる．米国でApple Iが登場し，日本では開設されたばかりの秋葉原のBit-INNにTK-80を求めるマニアが詰めかけ，マイコンという言葉が使われるようになった年だ．その年の春に，トムキャットコンピュータは8080を使ったマイコンキットBIG-1を作り，AMD社を通じて販売していた．20万円弱のBIG-1は，その後に出た8万円台のTK-80の前に姿を消した．同社はその後，S100バスやマルチバスのシステムやボードを手がける時期を迎える．こうしたシステムは，「現在のパーソナルコンピュータというより，ミニコンの代わりにデータロガーなどに使われた特注品」だったと，同社の佐々木明社長は話してくれた．

「そのころから，コンピュータはコンパチブルでなければならないという信念を持っていました．当時もインテルをはじめナショナルセミコンダクタ，モトローラなどの会社がそれぞれCPUを作っていましたし，日本のメーカーも独自のCPUを持っていました．私どもは，電卓からスタートし，電卓のROMの設計をやっていました．そうするとCPUが変わるたびにROMもやりなおしていました．コンパチブルであればそうした効率の悪いことをやらなくてすむわけです」と，佐々木氏は当時を振り返る．

遅れをとったIBM PC互換機 AT互換は世界最初を実現

1981年，IBM PCが発売された．トムキャットコンピュータはすぐにその互換機を開発した．「この互換機はDOSレベルの互換性しかもっていま

ワンラインNEWS
日本電気は1.8インチのHDD「D1711」を発売した．1.8インチHDDの発売は国内ではじめて．これまでの2.5インチに比べ体積，重量をともに1/3とした．

せんでした．コピーライトの問題がありますから，BIOSレベルの互換機まではできないんじゃないか．当時はそう思っていたのです」(佐々木氏)．ところが，やがて米国ではほぼ完全な互換性を有したPCが次々と登場することになる．

そこで，1984年にIBM PC/ATが発表された時には，早くから互換機の開発にとりかかり，同年末には世界で最初のPC/AT互換機を完成させた．これは，製品として世に出ることはなかったが，そのノウハウが米国や韓国のメーカーに販売され，事実上の世界標準機となるPC/ATの普及にひと役買うことになる．

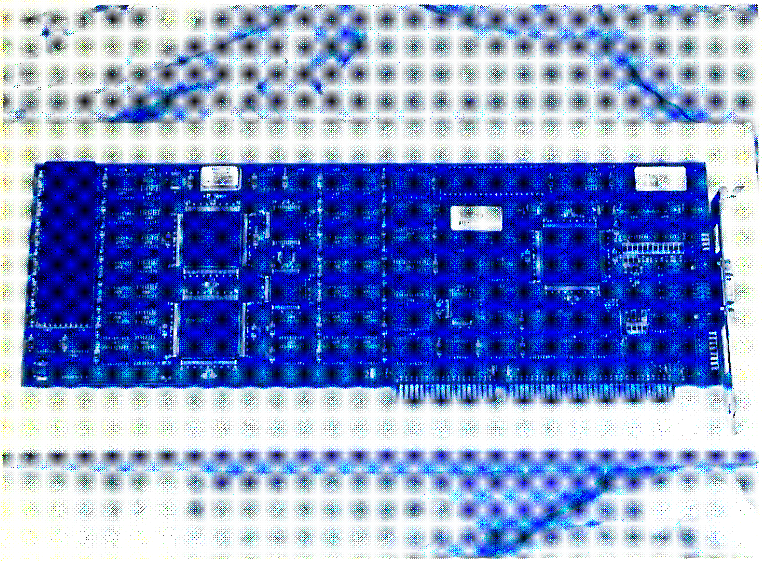
そのトムキャットコンピュータが，日本で優勢な日本電気製のPC-9801互換機を開発したのは1986年のことだった．このマシンもそのものは発売されることはなかったが，エプソンなどのメーカーに技術ノウハウが供与された．翌87年のエプソンのPC-98互換機の登場に結びつくことになる．

さらに，1987年春には，複数の異機種との互換を可能にする仮想システム論理回路（VSL）を開発し，それが川鉄によってLSI化された．VSL技術は同年にパソ協から発表されたダブル互換機に，そして1990年に発売されたASTリサーチのダブル互換機に取り入れられた．しかし，「思った以上にAT互換機の需要が少なく」(佐々木氏)，製品の販売は伸びなかった．

DOS/V，Windowsの登場で やっとATアーキテクチャが普及へ

こうして，トムキャットコンピュータの歴史を振り返ってみると，ATバスのボードとソフトウェアによって仮想98を実現するVirtual-98という製品が登場してきたのは当然のように思える．それにもかかわらず，PC/ATがごく限られたユーザーにしか使われていなかった日本では，少し違った受け止め方をされたようだ．

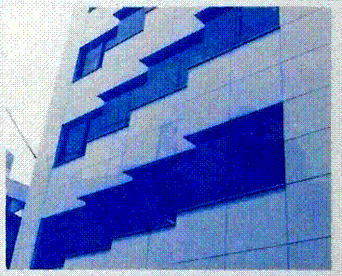
最初に98互換機を開発した会社として，次にIBM PC，PC-98両互換機を開発した会社として，トムキャットコンピュータは知られるところとなっ



XDC-IIボード

About Company

現在，社員数は約20名．本社は東京・荻窪，技術開発センターを北海道の旭川におく．ハード・ソフトに両方にわたるシステム開発を行っているため，開発にはハードウェアの知識が不可欠．新入社員は半田ゴテを持つところから教育されるという．



Products

・Virtual-98

PC/ATマシンでPC-9801のソフトを実行するための製品．ATバスの拡張ボード（XDC-II）とソフトウェア（V98）で構成される．拡張ボードは，PC-9801のディスプレイ機能とシステムクロックが搭載されている．ソフトウェアは，386，486の仮想86モードとプロテクトモードで動作する．必要なハードウェア条件は次の通り．

・CPU386SX以上のPC/ATマシン
・メモリ1.5MB以上
・フルサイズのATボードが入る拡張スロット
・マルチスキャンのCRT

CRTディスプレイは，98モードの時はXDC-IIのコネクタに，VGAなどPC/ATのビデオボードを使うときはそのコネクタに接続する．切り替えは簡単なスイッチャーを用意しておくと便利だ．
動作の順序は次のようになる．

①DOS/VなどPC/ATのMS-DOSを起動する
②V98を実行する
③V98モードになりオープニング画面が出る
④98のMS-DOSが読み込まれる
⑤98のアプリケーションを実行する

98モードのままリセットするために，仮想リセットキーが設定されている（[Alt]+[Ctrl]+[Ins]）．仮想98のディップスイッチ，メモリアドレスなどの設定を行うためのユーティリティが用意されている．
動作確認はMS-DOSビジネスソフトを中心に進められている．一太郎，花子，松，Z's STAFF kid98など多くが動作確認されている．現在，動かないと確認されているものに98用のWindowsがある．また，変則的なプロテクトがかかっているゲームソフトも動かない可能性があるという．
98モードからPC/ATモードへの切り替えはリセットが必要だが，ソフト的に切り替えができるバージョンを開発中．
価格：Virtual-98　120,000円
評価システムEX2000　698,000円

た．PC-98互換機が基本にあり，それにAT互換が追加されたという印象があった．日本のユーザーにPC/ATアーキテクチャという概念は存在しなかったのである．奇しくも1990年秋にASTリサーチのダブル互換機が発表されたのと同時期に登場したDOS/Vによって，日本のユーザーはPC/ATアーキテクチャーに目覚めることとなった．しかし，本当にATバスのボードが広く利用されるようになったのは，海外のビデオボードを日本語Windowsで動作させることを可能にしたDDD（ディスプレイディスパッチドライバ）が登場した今年の春からのことである．

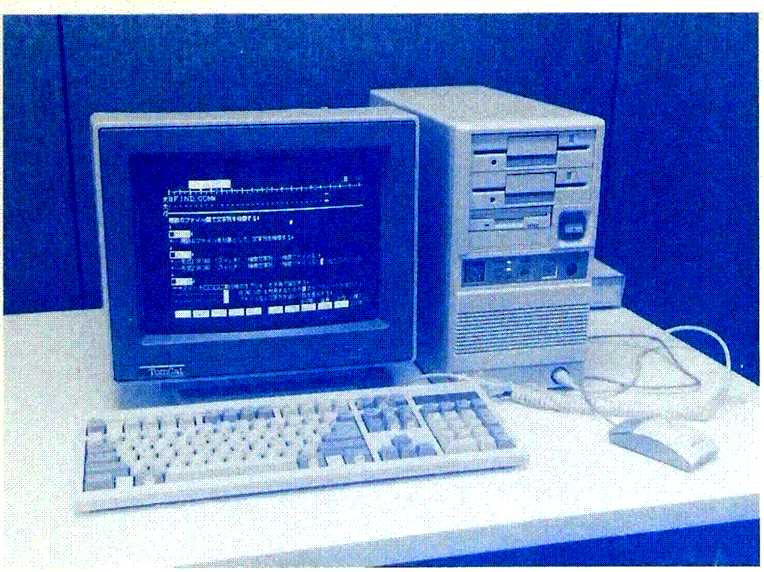
Windowsの普及は3年後 それまでは98ソフトが有効

Virtual-98には，4月20日の朝日新聞の記事になって以来，多くの問い合わせがあるという．ソフトハウス，ディーラー，個人ユーザーなど内容は多岐にわたっている．ソフトハウスは，DOS/Vへの移植を行わなくても自社のソフトがDOS/Vマシンで動くのでないかという期待があり，ディーラーは製品を扱いたい，そしてPC-98のソフトをDOS/Vマシンで動かしたいという個人ユーザーなどそれぞれの思惑がある．

「日本でWindowsが普及するのは3年先とみているんです」と，佐々木氏は話す．「それまでは98のソフトを動かさなければ，DOS/V自体も普及はむずかしいのではないか．それに，ノートブック型パソコンはこれからも需要があるでしょうが，これはノーマルの解像度で使えばいいものです．ソフト会社の負担を減らし，ユーザーの資産を生かす意味でもVirtual-98は活躍できると思います」（佐々木氏）．

将来的には，「デスクトップ機の1ウィンドウに98のノーマル画面を表示し，ノートパソコンとのコミュニケーションを可能にしたい」（佐々木氏）という．DOSレベルのウィンドウではなく，ハードウェアボードをも使うことにより，BASICや他のOSも動作する完全な仮想マシンを実現することをテーマにしている．

Virtual-98の供給方法は，まだ定まっていない．「ボードとソフトのみで供給すると，必ずサポートの問題が発生します．うちのような小さい会社ではパンクしてしまうことが目に見えています」と話すのは，同社経営企画室の牛尾英勝氏．期間を決めてインストールサービスを実施することを検討している．また，評価用システムとして，486/33MHzのPC/AT互換機（メモリ8MB，200MBHDD）にVirtual-98，ディスプレイなどをセットしたマシンを販売している．


松 5.0が動作している

ワンラインNEWS
キヤノンは印字の高速化など機能を強化したインクジェットプリンタ「バブルジェットプリンタBJ-15V」を発売した．従来機に比べ印字速度を1.3倍に高めた．

●イベント

■7月21日(火)～7月23日(木)

ネットワーク'92

LAN, WANなどのネットワーク関連機器とアプリケーションの展示会, コンファレンスも同時開催される.
会場：日本コンベンションセンター（幕張メッセ）
入場料：1,000円
問い合わせ先：(社)日本パーソナルコンピュータソフトウェア協会　TEL 03-3221-7481

■8月26日(水)～8月28日(金)

SunWorld Expo/Tokyo

サン・マイクロシステムズ社のSPARCstation関連製品の展示とコンファレンスが行われる.
会場：日本コンベンションセンター（幕張メッセ）
入場料：【コンファレンス・展示会3日間共通】35,000円（前売り）・38,000円（当日），【展示会3日間共通】2,000円（前売り），【展示会1日入場券】1,500円（当日）
問い合わせ先：IDGワールドエキスポ／ジャパン SunWorld事務局　TEL 03-5276-3751

●セミナー

Microsoft University

■7月20日(月)～7月21日(火)
8月24日(月)～8月25日(火)

C言語入門

Quick Cを使ったC言語の入門コース.
受講料：60,000円（2日間）

■7月22日(水)～7月24日(金)
8月26日(水)～8月28日(金)

C言語応用

MS-C 6.0を使ってC言語の高度な技術を学ぶ実習中心のコース.
受講料：90,000円（3日間）

■8月3日(月)～8月7日(金)

Microsoft Windows Programming Environment

Windowsアプリケーション開発のための入門コース.
受講料：180,000円（5日間）

■8月18日(火)～8月21日(金)

MS OS/2 Programming Environmet

MS OS/2アプリケーション開発のための入門コース.
受講料：150,000円（4日間）

■8月24日(月)～8月28日(金)

Microsoft Windows Applications Programming

Windowsアプリケーション開発のための上級コース.
受講料：180,000円（5日間）

Microsoft Universityは, OS/2, Windowsアプリケーション開発者のために定期的に開催されているセミナーです.
問い合わせ先：富士ソフトウエア(株)品川教育センター
TEL 03-5461-8296

NEWS & NEW PRODUCTS

アップルコンピュータ，QuickTimeスタータキットほかを発売

アップルコンピュータ（株）（TEL 03-3592-0505）は，MacintoshのOSの機能を拡張し，音声，ビデオ，アニメーション，圧縮された静止画像を簡単に使うことのできるソフトウェア「QuickTimeスタータキット（英語版）」を発売した．QuickTimeをサポートする製品を用い，テキストやグラフィックスを扱うように，ドキュメントやプレゼンテーション資料にQuickTime Movieをコピーして使うことができる．同製品には，QuickTime Movieを再生する「MoviePlayer」，静止画像ファイルを圧縮・伸長する「Picture Compressor」などの4つのユーティリティとサンプル資料が含まれている．価格は25,000円．このほか，MS-DOSのディスクを読み書きできる「Macintosh PC Exchange」（価格：15,000円），Macintosh用UNIX OS「A/UX 3.0（英語版）」も発表した．

また同時に，CPUに68040（33MHz）を搭載した，Macintoshの最上位機「Macintosh Quadra 950（英語版）」（価格：1,438,000円～1,838,000円）と従来機「MacintoshLC」を機能強化した「MacintoshLCII」（価格：258,000円～378,000円）も発表した．

USL，ビジネス市場をターゲットとしたSVR4.2を発表

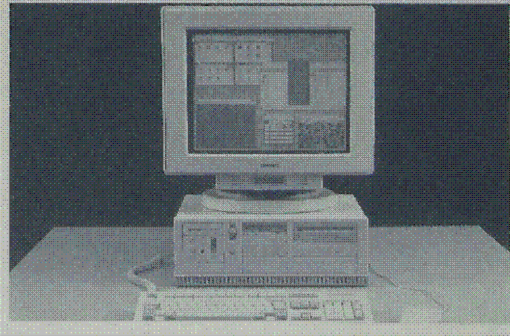
UNIXシステムラボラトリーズ（USL）は，UNIX System Vリリース4.2（SVR4.2）を発表した．SVR4.2は，32ビットビジネスパーソナルコンピューティングを標榜しており，GUIによるシンプルな操作と軽いシステムを実現している．最小限のシステム構成の場合，386SX/16MHz，4MBのメモリ，60MBのHDDで動作する．ドラッグ・アンド・ドロップの操作，DOS・Windowsアプリケーションへの対応，アドビタイプマネージャの組み込みなどビジネスPC市場を意識したシステムになっている．ルック・アンド・フィールはOSF/MotifとOPEN LOOKが選択できる．CPU別のポーティングは，まずインテルが優先され，次いでSPARC，MIPS，来年にはRS6000，DEC Alpha，HP/PAへの移植が予定されている．エンドユーザ一向けのシュリンクラップSVR4.2は，ノベル社とUSLの合弁会社UNIVELから提供される．また，日本語対応は今年後半の予定．

日本オリベッティ，ACE/ARC仕様準拠RISCマシンを発表

日本オリベッティ㈱（TEL 03-3714-1222）は，ACE InitiativeのARCSに完全準拠したパーソナルコンピュータ「Olivetti RISC M700-10」を発表した．R4000 RISCプロセッサを搭載し，「Windows NT」と「ACE UNIX SVR4」の2つのOSを利用することができる．高解像度ビデオコントローラ，イーサネット，オーディオ入／出力などの各種インターフェース，4つのEISAバススロットを備えている．17インチカラーディスプレイ，キーボード，マウスが付属した標準構成は1,750,000円，同社が認定したソフトウェア開発者には，16MBのメモリを搭載したディベロッパーズ・システム（910,000円）が用意されている．

■おもな仕様

プロセッサ：MIPS R4000PC（50MHz）／メモリ：8MB（最大256MBまで拡張可能）／外部記憶装置：2.88MB 3.5インチFDD×1・210MB HDD×1・550MB CD-ROM×1／拡張スロット：EISAスロット×4／価格：1,750,000円（17インチカラーディスプレイ，キーボード，マウス添付）



ワンラインNEWS
富士ゼロックスは解像度が1インチ当たり600ドットのレーザプリンタ「富士ゼロックス4109HL」を発売．「PostScript」搭載，高品位印刷を可能にした．

NEWS & NEW PRODUCTS

エプソン・日本DEC，ATアーキテクチャサーバをダブルブランドで発売

セイコーエプソン（株），エプソン販売（株），日本ディジタルイクイップメント（株）の3社は，コンピュータ市場の流れであるPCインテグレーション市場，PC-LANシステム分野への本格的取り組みを目的とした業務提携による活動を開始，その第一弾製品として，「DECpc400サーバ・シリーズ」を「digital」，「EPSON」のダブルブランドで商品化，エプソン販売を通じて販売する．

「DECpc400サーバ・シリーズ」は，PC/ATアーキテクチャを採用，80486（SX・DX）を搭載している．また，現在セイコーエプソンが日本語化を進めている「PC-UNIX（日本語SVR4）」（1992年12月発売予定）を搭載することができ，NetWareなどのネットワークOSもサポートしている．CPUボードの交換によるアップグレードが可能，メモリは標準で4MB，最大192MBまで拡張が可能，CPU，メモリ間とメモリI/O間の性能を大幅に向上させる高速CPU技術「X-Press EISAバス高性能モジュラ構造」の採用により，高速処理を実現している．

コンパクトな普及型ミニタワータイプの「DECpc400STサーバ・シリーズ」と拡張性を強化したデスクサイド型の「DECpc400xPサーバ・シリーズ」の3機種6モデルがある．価格は基本最小システム構成で，1,050,000円から．

日本IBM，コンパクトで，低価格なOS/2マシンを発売

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，ディスプレイ一体型で省スペースタイプのPS/55シリーズの新モデルとして，「PS/55 モデル5530-W」と「同モデル5530-V」を発表した．両モデルともOS/2 J2.0導入済み製品が用意されている．

■PS/55モデル5530-W0A，WYA，WJA

プロセッサ：80486SX（20MHz）／メモリ：4MB（W0A/WJA，最大28MBまで拡張可能），8MB（WYA，最大32MBまで拡張可能）／外部記憶装置：3.5インチFDD×1・100MB HDD×1／拡張スロット：マイクロチャネル32/16ビットフルサイズ×2／価格：1,020,000円（W0A），1,130,000円（WYA，OS/2 J2.0導入済み），1,060,000円（WJA，OS/2 J2.0導入済み）

■PS/55モデル5530-V04，V08，VY8，VJ8

プロセッサ：386SLC（20MHz）／メモリ：2MB（V04），4MB（V08，VJ8），8MB（VY8）（各モデルとも最大16MBまで拡張可能）／外部記憶装置：3.5インチFDD×1・40MB HDD×1（V04），3.5インチFDD×1・80MB HDD×1（V08，VY8，VJ8）／拡張スロット：マイクロチャネル16ビットフルサイズ×2・ハーフサイズ×1／価格：780,000円（V04），880,000円（V08），990,000円（VY8，OS/2 J2.0導入済み），920,000円（VJ8，OS/2 J2.0導入済み）

関西電機，486DX2 50MHz搭載のDOS/Vマシンを発売

関西電機（株）プラスワン（TEL 03-5687-6912）は，CPUに80486DX2（50MHz）を搭載したDOS/Vパソコン「P1-48AH」を発売した．メインメモリ4MB，内部キャッシュメモリ8KB，外部キャッシュメモリ256KBを搭載，CPU速度（従来機48AGの約1.5倍）以外は，従来機とほぼ同一の仕様となっている．添付ユーティリティソフトとして，日本語Windows用の高解像度ドライバや一太郎・花子などをバスマウスで動作させるためのPS/2マウスシミュレータソフトなども標準添付している．

■P1-48AH-STD/P1-48AH-H120

プロセッサ:80486DX2（50MHz）／メモリ：4MB（最大32MBまで拡張可能）／外部記憶装置：3.5インチ・5インチFDD×各1（STD），3.5インチ・5インチFDD×各1・120MB HDD×1（H120）／拡張スロット：ATバス×8（内ユーザースロット4）／価格：698,000円（STD），798,000円（H120）

NEWS & NEW PRODUCTS

横河・ヒューレット・パッカード，世界最小1.3インチHDDを発表

横河・ヒューレット・パッカード(株)(マスストレージマーケティングセンタ：TEL 03-3335-8315)は，ヒューレット・パッカード社が開発した世界最小1.3インチハードディスク駆動装置「HP Kittyhawk」パーソナル・ストレージ・モジュール(PSM)を発表，日本国内で販売を開始する．「HP Kittyhawk」は，マッチ箱ほどのサイズ(厚さ10.5ミリ，幅36.5ミリ，奥行50.8ミリ)で，21.4MBの容量をもち，重さも32グラムと世界最軽量，動作時100グラムという耐衝撃性を実現した設計となっている．ICカードと同じPCMCIAインターフェースをサポートしており，着脱可能なメモリモジュールとして使用することもできる．サンプル価格は64,800円で，OEMチャンネルで販売される．

京都マイクロコンピュータ，アドインアクセレータボードを値下げ

京都マイクロコンピュータ(株)（TEL 075-953-0963)は，従来より発売していたアドインアクセレータボード「Turbo-486シリーズ」の価格を改定，発売1周年にあたる7月から新価格での販売を行う．「Turbo-486シリーズ」は，80386を搭載したPC-9800シリーズを486マシンに性能アップするアドインアクセレータボードで，倍速クロックモードで40MHzの動作が可能（本体クロック20MHz時）な「Turbo-486DX40」は298,000円（旧価格398,000円)，Turbo-486DXは198,000円，Turbo-486SXは98,000円に，それぞれ価格改定を行った．

亜土電子工業，55noteで98用CRTを利用可能にするドライバソフト発売

（株）亜土電子工業（IBM営業部：TEL 03-3257-2755)は，IBM PS/55noteでPC-9800シリーズ用アナログディスプレイを利用可能にする「VGA/98ドライバー」を発売した．PC-9800シリーズ用640×400ドットのアナログカラーディスプレイで，VGA規格の640×480ドットの画面をカラーで表示，DOS/Vの日本語表示，DOS/Vの英語モードでのCGA，EGA，VGAのすべての表示が可能，VGA出力を内部で自動的にPC-9800用CRTディスプレイに表示可能なように変換するため，アプリケーションは変更なくそのまま使用することができる．価格は14,800円（ケーブル付き）．

エム・ピー・テクノロジー，Windows対応音声再生ソフトほかを発売

（株）エム・ピー・テクノロジー（TEL 03-3358-9800)は，2種のWindows 3.0対応の日本語版アプリケーションソフトを発売した．サウンドボードなどのハードウェアの追加を一切必要とせず，手軽に音声を再生できる「Sound Patio（サウンド・パティオ)」とアイコンやカーソルを好みのデザインに変えることができるユーティリティソフト「Icon Do-It(アイコン・ドゥ・イット)」．「Sound Patio」は，アメリカでは「Sound Palette」という名称で発売されており，米国BERKELEY SYSTEMS社のWindows用スクリーンセーバー「After Dark 2.0」にも，その機能が付加されている．サウンドのフック設定，アラーム設定が可能，Excelなどのマクロを使って簡単に音声再生ボタンを作ることができる．また，開発者向けにDLL(ダイナミックリンクライブラリ）を同梱している．PC-9800シリーズ用（スタンダードモードのみ）とDOS/Vマシン用があり，価格はいずれも9,800円．「Icon Do-It」は，米国Moon Valley Software社の製品で，200種のデザインアイコンと15種のアニメーションアイコンが用意されている．日本語版には，日本語DOS用ソフトのアイコンなどエム・ピー・テクノロジー社オリジナルデザインのアイコンも用意されている．価格は9,800円．

ワンラインNEWS
台湾の大手電気メーカーである大同公司の日本法人である大同日本は，日本でのパソコン事業を強化する．OADGに加盟，国内で低価格なPC互換機を発売する．

NEWS & NEW PRODUCTS

エクセルソフト，Win対応，米国hDC社製品の日本語対応版を発売

エクセルソフト（株）（TEL 03-5440-7875）は，MS-Windows 3.0に対応した米国hDC社のアイコン作成ツール「hDC Icon Designer」の日本語版を7月から発売する．同社は，hDC社の国内販売代理店として，すでに「hDC Power Launcher」などの英語版を販売しているが，現在，hDC社の全プロダクツの日本語化を進めており，「hDC Icon Designer日本語版」は，その第一弾．同製品は，アイコン作成のための11個のボタン道具箱を装備し，簡単にオリジナルアイコンを作成することができる．作成したアイコンはプログラムマネージャ上のアイコンメニューにも利用することができる．200種類のサンプルアイコンが用意され，BMP，GRPなどのファイル形式のデータを読み込んで編集することができる．価格は12,800円．

住友金属工業，技術計算ソフトMathcadのWindows対応版を発売

住友金属工業（株）（システムエンジニアリング事業本部：TEL 03-5476-9816）は，技術計算ソフトでは世界のトップシェアを誇るMathcadのWindows対応版「Mathcad 3.1 for Windows」を発売した．これまでの技術計算用ソフトウェアと違い，プログラミング言語やスプレッドシートを必要とせず，数学表記での入力で動作する．シンボリック計算に対応，オプションで高等数学から化学工学までの高度な計算のための11種のアプリケーションパックを用意，解析ですぐ使用できる「立体の体積」，「発振器の共振周波数」などの数百の式，定数，グラフが盛り込まれ，対話形式で簡単にアクセスすることができる電子ハンドブックにも対応している．年内には日本語バージョンの発売も予定している．価格は128,000円で，UNIX版（198,000円）もある．

IBM，松下電器産業とBIOSライセンス契約を締結

IBMコーポレーションは，松下電器産業（株）との間で，ATバス搭載のPCオープン・アーキテクチャー準拠のBIOSに関し，ライセンス契約を締結したと発表した．この契約締結は，IBMが全世界的に推進しているオープン・システムの構想の一環として行われたもので，従来，日本アイ・ビー・エム（株）が，PCオープン・アーキテクチャー推進登録企業に対して行っていたBIOS外部インターフェースの開示に加え，はじめてBIOSコードそのものをライセンス開示したもの．同社では，これによりPCオープン・アーキテクチャーに準拠し，短期，低コストで開発されたBIOSを搭載したパソコンの製造販売が容易になったとしている．

第1回「フリーソフトウェア大賞」決定，「LHA」が大賞授賞

個人の使用や流通の自由を認めた「フリーソフトウェア」は，現在，パソコン通信，ネットワークサービスなどで約10万本もの作品が流通している．こうした背景をベースにネットワーク・サービス事業者，コンピュータ雑誌社の有志により「FSP'92（Free Software Prize 1992）」のプロジェクト名のもと「フリーソフトウェア大賞実行委員会」（フリーソフトウェア大賞運営事務局：TEL 03-3233-0965）を組織，5月中旬にかけて優秀作品を選び，その作者に対して表彰を行った．大賞は「LHA」，選考委員会特別賞は「ish」が授賞，ジャンル部門賞では，通信ソフトウェア部門賞は「WTERM」，ユーティリティ部門賞は「BDIFF」，「DIET」，「FD」，「LV」，「xscript」，アミューズメント部門賞は「Super Depth」がそれぞれ授賞した．

Programmer's Program
ディスク容量倍増ツール
DiskXII

購入してしばらくの間は，ハードディスクドライブ（以後HDD）には使い切れない容量があるように感じる．ところが，半年から1年の間に「残り容量もあとわずか……」なんて状況に陥っている人が大半であろう．HDDの価格は年々下がってはいるが，それでも決して安いものではない．不要なファイルを削除したり，あまり使用しないファイルをフロッピーディスクに移動したりしたところで，フリーエリアはほとんど増えない．こんな時，「HDDの容量を増やしてくれるような都合のいいソフトウェアがあれば」なんて考えたことはないだろうか．そんなソフトウェアがついに登場した．

「DiskXII」は，ファイルでいっぱいになってしまったHDDの容量を倍増し，空き容量を作り出すデバイスドライバ形式のソフトウェアである．

簡単なインストール

インストールプログラムで設定を行い，再び立ち上げるだけで「圧縮ドライブ」が生成される．あとは従来のドライブと同様に操作を行うことができるようになる．また圧縮ドライブの設定変更や解除もインストールプログラムから簡単に行うことができる．

圧縮ドライブの設定はHDDの空き領域を利用して作成する方法（図1）と，すでに使用している領域を圧縮ドライブとする方法（図2）がある．これもインストール時に指定するだけ，すべてインストールプログラムが自動的に行ってくれる．

たとえばA・BドライブがHDD，C・DドライブがFDDの環境で，Bドライブの領域を使って圧縮ドライブを作成した場合，圧縮ドライブはEドライブとなる．さらにドライブの位置を入れ換えることもインストールプログラムから設定可能で，EドライブとBドライブを入れ換えることで，ドライブ名を変更することなく容量を増加させることが可能である．

なお設定を変更した場合，リセット後その変更内容が反映されるため，自分でリセットスイッチを押す必要がある．インストールプログラムがリセットをかけてくれない点が少し気になった．

圧縮ドライブ

DiskXIIはデバイスドライバ形式のプログラムである．インストールプログラムによって起動ドライブのCONFIG.SYSに

```
DEVICE=DISKX.SYS
```

の1行を追加，指定したドライブから領域を確保し，それを「圧縮ドライブ」と呼ばれる仮想ドライブとして追加す

図1

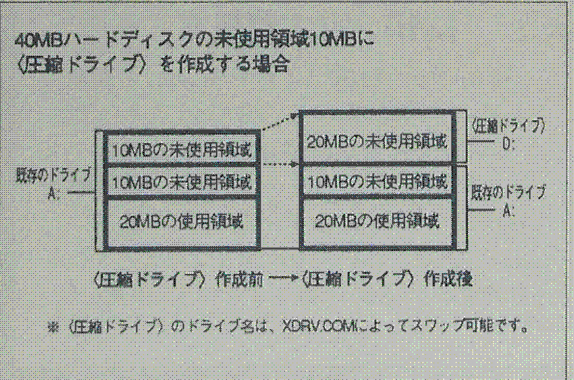
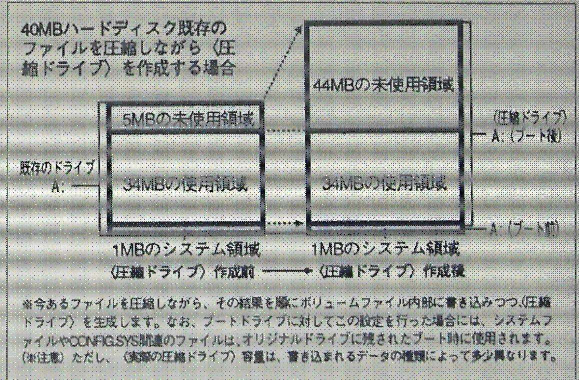


図2



ワンラインNEWS
横河・ヒューレット・パッカードは米ヒューレット・パッカードが開発した1.3インチ型HDDを国内で販売する．

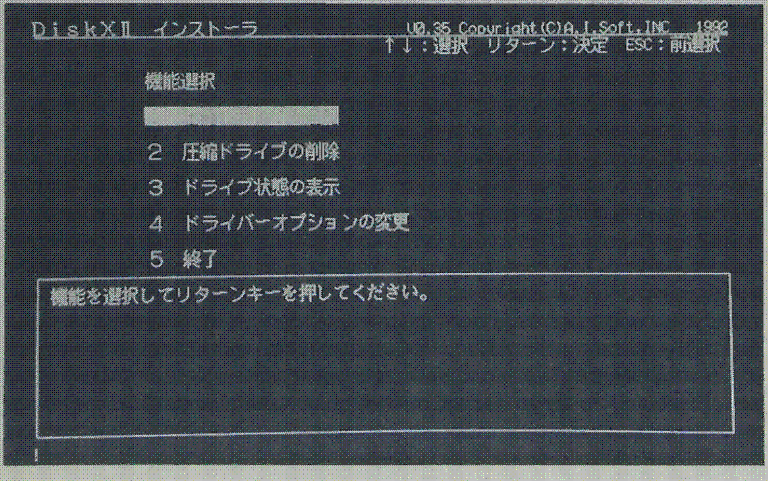
る．もちろん，この圧縮ドライブは，DOSから見れば通常ドライブとなんら変わることなく扱うことができる．

圧縮ドライブはMS-DOSのコマンドラインからの操作はもちろん，各種ファイル管理ユーティリティやバックアップユーティリティからも通常のドライブ同様にアクセスすることができる．またDiskXⅡに付属するユーティリティを使うことで圧縮率を知ることもできる．

圧縮展開作業を意識しない

圧縮ドライブにアプリケーションソフトウェアをコピーすると，プログラムはDISKX.SYSによって自動的に圧縮されて書き込まれる．またアプリケーションプログラムを実行すると，やはりDISKX.SYSによって自動的に展開されて実行される．同様にしてアプリケーションプログラムが作成するデータファイルも，アプリケーションから読み書きされるときに自動的に圧縮展開される．したがって，圧縮展開作業をまったく意識することなく通常のドライブと同様に利用できる．さらにファイルの一部を読み書きする場合でも，自動的に必要な部分のみが圧縮展開されるので，データベースファイルや日本語FEPの辞書ファイルに対しても有効である．もちろん，この場合でも圧縮展開作業を意識する必要はまったくない．つまりファイルの圧縮展開作業において，コマンドや複雑なパラメータを指定する必要は一切なく，すべてのファイルを圧縮展開することが可能となる．

圧縮率はデータの種類によって異なるが，平均して50%程度だった．つまり圧縮ドライブには，今までのディスク容量の約2倍のデータが格納できるようになる．当然，圧縮ツールで圧縮したファイルに対してはほとんど効果が見られないが，普段から利用するアプリケーションやデータを従来の圧縮ユーティリティで圧縮していることはまずあるまい．DiskXⅡは従来の圧縮ユーティリティとは違い，ファイルのサイズを小さくすること自体が目的なのではなく，ドライブの容量を増大させるためのユーティリティである．



◀β版のインストーラー

■おもな特徴

- 自動的に圧縮展開を実行するので，コマンド入力が不要
- ファイル全体を展開する必要がない
- 展開用の領域を確保する必要がない
- ファイルを圧縮したままの状態でアクセス可能
- MS-DOSでディスクアクセスするすべてのソフトウェアで利用可能（ただしMS-Windows 3.0のエンハンスドモードにおける恒久スワップファイルは除く）
- メニュー指示形式で簡単なインストールプログラム
- 圧縮比率の確認やパスワードの設定が可能

■DiskXⅡ

販売：エー・アイ・ソフト（株）（TEL 0424-85-7444）／価格：12,800円／対応機種：日本電気PC-9800シリーズ（ただしVM2以降）およびセイコーエプソンPC-286/386シリーズ／対象ドライブ：ハードディスク，RAMディスク／動作OS：MS-DOS 3.x, 5.x, MS-Windows 3.0／必要メモリサイズ：約35KB（ただしEMSメモリ使用時は約4KB），MS-DOS 5.0のハイメモリロード（DEVICEHIGH=）に対応／対象ソフトウェア：MS-Windows 3.0のエンハンスドモードの恒久スワップファイル以外のMS-DOSでディスクアクセスを行うすべてのファイル（.COMや.EXEを拡張子に持った実行ファイルやオーバーレイファイルを含む）

ワンラインNEWS
日本電気と日本アイ・ビー・エムは両社の汎用コンピュータを連動させるソフトを共同開発。それぞれ営業活動を開始した。

オンラインソフト採集帳 20
Hyper mmm wtmmm

Hyper-mmmとは

mmmは，フリーソフトウェアのパソコン用多回線BBSホストプログラムである．このホストプログラムはハイパーノーツと呼ばれる優れた操作性を持ったシステムである．このハイパーノーツは多機能で便利だが，ついつい接続時間が長くなりがちである．とくに遠距離の場合は電話代もばかにならない．メッセージを一気に読みだしておいて，そのログファイルをエディタやページャーで眺めるのもよいが，メッセージの検索もたいへんだ．なにか便利なツールはないものか……．

WTERMなどの通信端末ソフトで記録した，mmmを使ったBBSのログファイルから，オフラインでハイパーノーツ機能を実現する，それがHyper-mmmである．

起動の準備

HMMM03ZD.LZHを復元して生成されるファイル群から次の3つのファイル

```
hmmm.exe
hmmm.cfg
hmmm.hlp
```

をパスが通ったディレクトリにコピーする．もちろんページャーやエディタもパスの通ったディレクトリに用意しておく必要がある．

とりあえず，これで起動可能となる．MS-DOSのコマンドラインから

```
C> hmmm <ログファイル名>
```

と入力すれば，とりあえずはハイパーノーツの世界である．

ただし，このままでは単なるログファイルの閲覧ツールである．hmmmは起動時に同じディレクトリに存在する設定ファイルhmmm.cfgを参照する．このファイルにはhmmmの動作に関する設定と，各ネットに関する設定を記述する．内容についてはHMMM03ZD.DOC中に詳しく解説されているので，自分の環境および自分のアクセスする各BBSにあわせて変更する．また，オンラインでのメッセージ・メールの読み書きを代行してくれるwtmmmなども自分の環境および各BBSにあわせて設定すれば，さらに便利になる．

使ってみる

設定が終わったら，通信ソフトを立ち上げ任意のBBSにアクセスしてログファイルを作る．WTERMの場合はオートログイン文字列でマクロの設定をしておけばアクセス先を指定するだけで，新規メッセージおよびメールを読んで，ログアウトまで自動運転できる．

あとはMS-DOSのコマンドラインから

```
C> hmmm
```

と入力するだけである．まずメニューが表示されhmmm.cfgで設定したBBSからアクセス(?)するBBSを選択する．

```
MAIN>
```

と表示され，オフラインでログイン完了である．もちろんhmmmの操作はmmmのコマンドモードに準拠している．あとは複数のログファイルを気にすることなく，しかもどんなに長時間読んでも書き込んでも，電話代はかからない．またhmmmから書き込んだメッセージやメールは，次回のアクセス時にwtmmmが自動的に処理してくれる．

●特徴

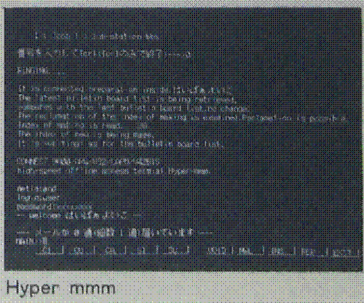
・mmmを使ったBBSのログファイルから，オンラインと同様の手順でメッセージを読むことができる．

・単なるログファイルの閲覧だけでなく，複数のログファイルを有機的に処理できる．

・wtmmm（WTERM用マクロ）などと併用することで，Hyper mmm上でメッセージの書き込み，電子メールの送信ができる．

・メッセージやメールの書き込みは使い慣れたエディタでスクリーンエディットできる．

・テグレット技術開発より発売されている非定型データベースプログラム「知子の情報」に登録可能な知子テキス



Hyper mmm

ワンラインNEWS
インターナショナル・サイエンティックはアップルコンピュータの「MacintoshIIシリーズ／Quadraシリーズ」に搭載する16MビットのSIMMを発売

トにログファイルを変換できる.

■Hyper mmm

名称：Hyper mmm／分類：通信ユーティリティ／登録名：HMMM03ZD.LZH／作者：Ichii.Ando（坪山博貴）／動作環境：MS-DOS ver.3.1以降の動作する機種／復元方法：LHA E HMMM03ZD

■wtmmm

名称：wtmmm／分類：WTERM用マクロ／登録名：WTMMM16D.LZH／作者：Ichii.Ando（坪山博貴）／動作環境：WTERMの動作する機種／復元方法：LHA E WTMMM16D

ファイル構成

```
  Name          Original    Packed  Ratio   Date       Time    Attr Type  CRC
 -------------  --------  --------  -----  --------  --------  ---- ----- ----
  README.DOC         245       241  98.4% 92-06-01 03:42:00 a--w -lh5- 51E1
  HMMMUPDT.DOC      7593      3469  45.7% 92-06-01 03:42:00 a--w -lh5- CAD0
  HMMM03ZD.DOC     61985     17726  28.6% 92-06-01 03:42:00 a--w -lh5- CC17
  HM3ZDIDX.DOC      6238      1546  24.8% 92-06-01 03:42:00 a--w -lh5- 35FE
  HM3ZDTAG.DOC      4214      1526  36.2% 92-06-01 03:42:00 a--w -lh5- 218A
  HMMM.EXE         74804     40674  54.4% 92-06-01 03:42:00 a--w -lh5- F588
  HMMM.CFG           360       140  38.9% 92-05-25 03:40:00 a--w -lh5- 537A
  HMMM.HLP          3118      1008  32.3% 92-05-25 03:40:00 a--w -lh5- 1988
  HMMMNOMS.COM     13205      9697  73.4% 92-06-01 03:42:00 a--w -lh5- FC71
 -------------  --------  --------  -----  --------  --------
     9 files      171762     76027  44.3% 92-06-01 22:27:24

Listing of archive : WTMMM16D.LZH

  Name          Original    Packed  Ratio   Date       Time    Attr Type  CRC
 -------------  --------  --------  -----  --------  --------  ---- ----- ----
  README.DOC         334       299  89.5% 91-12-01 00:16:10 a--w -lh5- 7C1C
  UPDATE.DOC        5536      2348  42.4% 91-12-01 00:16:10 a--w -lh5- EFD3
  WTMMM16D.DOC     43842     11253  25.7% 91-12-01 00:16:10 a--w -lh5- FA77
  NOT98.DOC          881       611  69.4% 91-12-01 00:16:10 a--w -lh5- CE0F
  NOTMNP.DOC         967       547  56.6% 91-12-01 00:16:10 a--w -lh5- F5FA
  USEWTP.DOC        4242       684  16.1% 91-12-01 00:16:10 a--w -lh5- 2B74
  3MMAIN.MAC        9653      2594  26.9% 91-12-01 00:16:10 a--w -lh5- C1C1
  3MPAD.MAC        14417      3378  23.4% 91-12-01 00:16:10 a--w -lh5- DC35
  3MUSER.MAC        1442       791  54.9% 91-12-01 00:16:10 a--w -lh5- 56D6
  CFGSAMP.LZH       4654      4654 100.0% 91-11-14 00:16:00 a--w -lh0- 4C18
 -------------  --------  --------  -----  --------  --------
    10 files       85968     27159  31.6% 91-12-01 00:16:10
```

オンラインソフト採集帳 番外編

斜め読み支援ツール Askance

現代社会では，多くの情報があふれている．このあふれる情報の中から自分に有益な情報のみを取得する必要がある．ペーパーメディアでは，「速読術」をマスターするための広告をよく見ることができる．ただし，この速読術はいまいち怪しげで，これをマスターした人の話は聞いたことがない．普通の人々は「斜め読み」によって，より多くの情報を断片的ながら得ているのである．

電子メディアでもペーパーメディア同様に，より多くのテキストファイルを読む必要性が出てきた．そこで秒単位で仕事をこなすスーパープログラマのための「斜め読み支援ツール」，それが「Askance」である．

AskanceはGDCを搭載した日本電気のPC-9801シリーズまたはその互換機上で動作する（ただし，有能なプログラマがはたしてこの機種を使用しているかは定かではない）．

MS-DOSのコマンドラインから

A> ASKANCE ASKANCE.DOC

と指定すればASKANCE.DOCを表示してくれる．Askanceは一種のテキストページャーである．テキストファイルの斜め読みとはどんな機能か，実際に動作させるのが一番であろう．ここでは詳しく触れないのでぜひ試してもらいたい．きっと納得していただけると思う．

■Askance

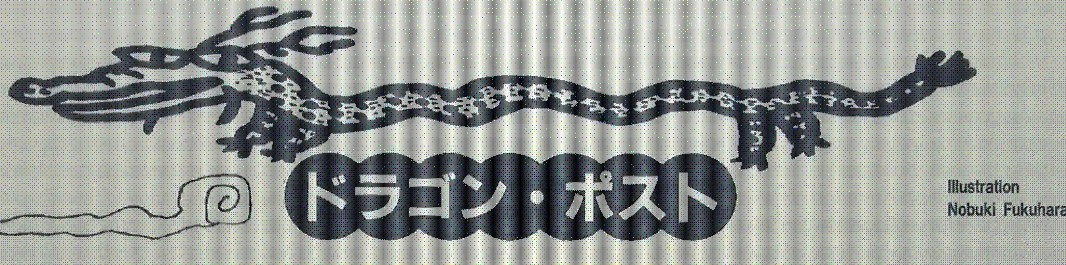
名称：Askance／分類：テキストファイルページャー／登録名：ASKANT12.LZH／作者：Youderng／動作環境：NEC PC-9801シリーズ（VM2以降）／復元方法：LHA E ASKANT12

ファイル構成

```
Listing of archive : ASKANT12.LZH

  Name          Original    Packed  Ratio   Date       Time    Attr Type  CRC
 -------------  --------  --------  -----  --------  --------  ---- ----- ----
  ASKSRC.LZH        4830      4830 100.0% 92-04-02 01:20:00 a--w -lh0- 7CEC
  ASKANCE.EXE       3096      1533  49.5% 92-04-02 01:20:00 a--w -lh5- 7840
  ASKANCE.DOC       3354      1983  59.1% 92-04-02 01:20:00 a--w -lh5- B49C
 -------------  --------  --------  -----  --------  --------
     3 files       11280      8346  74.0% 92-04-02 12:27:30
```

ワンラインNEWS
日本アイ・ビー・エムは「OS/2」にマルチメディア機能を付加する「MMPM/2」を発売した．MMPM/2はOS/2に各種AV機器の制御機能をもたせる．



ドラゴン・ポスト

Illustration
Nobuki Fukuhara

この欄は，読者からのお便りを中心に構成します．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．

★殺虫剤が欲しい!!

Ｃ言語を勉強し始めて約３カ月が経過しました．今，QuickCでプログラミングしていますが，虫だらけで手におえません．月刊プログラマーズ・ページ殿，よく効く殺虫剤を紹介して下さい．（パソ吉）

――虫は殺虫剤で殺しても，たいがいまた別のが出てくるものです．それに，殺虫剤を持って追いかけ回すのも骨の折れることです．虫が出る前に，整理整頓や掃除を心がけて清潔な環境を保っておくことをおすすめします．:-P

★夢はどこにある

最近，なんかおかしいですよ．10万円以下のV30マシンとMS-DOSで必要な事はほとんど出来るのに，何十万円も出して486マシンだの，Windowsだのって，なんか僕達の夢ってこんな所に有るのかなー?? もっともっと色んなシステムが出て来てもらわなきゃ，おもしろくないったらありゃしない．Tron頑張れ，Pen-Dos頑張れ，ETバンザーイ（^^;）(Xou)

――その通りです．16ビットのMS-DOSマシンで必要なことはほとんどできてしまいます．それを知っていても，Windowsもやがて変わってしまうとわかっていても，「欲しい」と思ってしまうのがパソコンの世界なのですね．

★こんどは大丈夫

Ｃでプログラムが組みたい，その思いで何冊の本を買っただろうか．しかし，すべて途中挫折しています．スーパービギナーにはむつかしすぎるんですね．でもこりないでまた買いました．「はじめて読むＣ言語」です．こんどはだいじょうぶだ!! かな？（橋本純一）

――スーパービギナーって，超初心者っていう意味なんですね．なんか，すごく優秀な初心者なのかと思ってしまった．（^^;）

★マシンと相談して……

私はマシンと相談の結果お互いに進化を止めてしまいました．よくもこれだけ次々と新機種が出て来ますね．コロコロと買い換えている人がいるのでしょうね，キット．CPUを80系と60系２つ持ったのが出ませんかね．人間も右脳と左脳が有るのに／（岩口正雄）

――WindowsNTとか，Soralisとか，32ビットのOSが出てくるとインテルのCPUでもRISCでも動くようになって，CPUの違いをあまり気にしなくてもよくなるかも知れない．あっ，でもやっぱりマシンを買わなくてはならないのか……．

★自覚症状はないんですが……

（アンケートハガキの）ユーザー区分にマニアというのがありますが，私は自覚症状がありません．だから一般ユーザーにしておいたんですが，個人ではX68000持ってて，実家に帰るとX1とAPPLE IIのこわれたのがあって，学校に勤めているんですが，学校にはSPARCと98がたくさんあります．これで自覚症状がないということは，もう手遅れでしょうか？（Mr.N）

――バグがありますと明示して売っているソフトがないように，「私はマニアです」とか「オタクです」と名乗る人もあまりいません．でも，Nさんの場合は，十分に自覚症状があるのにとぼけているというやつですね．

★プログラミングに挑戦／

アプリケーションソフトを使っていたのですが，最近パソコンでプログラムに挑戦してみようと思い立ちました．８年前，大学でPC-98でベーシック，研究室の大型コンピュータでFORTRANを使っていたのですが，今となっては役に立ちません．また，マシン語，アセンブラ，Ｃの知識はあまり無いため，何から始めようか迷っています．是非，貴誌の中でアセンブラ講座等のこれからプログラミングを始めようと思っている人に役に立つような特集を取り上げて欲しいと思います．（SHIMA）

――新しい言語を学ぶというのもいいですが，昔やったBASICやFORTRANを活用するという手もあると思います．バージョンアップされたり，Windows対応になったりで，まだ活躍しています．

★お願い／

ノートで動くC＋＋コンパイラがほしい．（天野守）

――話題になっているMS-C/C＋＋は，インストールするのに50MBほどのディスク容量が必要だとか．BC＋＋にしても同じようなもの．ということは……，C＋＋が問

題なく動くような強力なノートパソコンを探したほうがいいのでは？

★DOS/Vマシンの見積

DOS/Vマシンを買おうと思って（日本で）みつもってもらいました．スペックはジミに386DX（25MHz），8MB-RAM，80MB（IDE）HDD，3.5/5 FDD（各1），14" VGA CRT（カードも），387コプロ，トールタワーケース，マウス，モデム（2400，MNP5）＋GeoWorksProで……40万円でオツリがくる……．これではMac II siもアブナイし，98なんて……．STEPも危ない??（椎野倫人）

——パソコン関係の製品は，日本で作っているのに，海外のほうが安いというものが多くあります．CRTやメモリ，CD-ROMドライブ……などがそうでした．やっと，日本のパソコン関係製品も国際価格に近づいてきたのでしょうか．でも，STEPは，DOS/Vマシンをはじめたようです．

★パソコンか，WSか

パソコンとワークステーションの差がなくなりつつある昨今ですが，これからはどうなるのでしょう．まだまだパソコンが普及していくのでしょうか？　私はもうぼちぼちSUNなどのワークステーションにしようと思っているのですが，ところで貴誌は大変参考になります．これからもがんばってください．（中村朋朗）

——パソコンの機能がどんどん上がっても，ソフトは86時代のものがほとんどというのが，今の状況ですね．

★期待しています

インテルよりi486の倍クロックバージョンが出ました．386ユーザーには同じような倍クロックの386が互換チップメーカーから出ないものかなと，ひそかに期待しているのですが……．更に386とピン互換があって倍速クロックの486チップというのも出たらうれしいと思います．（田嶋広行）

——386ピン互換で，486並のスピードを持つ386互換CPUというのが出ているらしいです．CPUにしてもメモリにしても，部品の一つ一つが欲望を刺激してくれます．

★欲をいったら

PC-9801FAは確かに速いが，486では最低ランクの486SX（16MHz）というのはどうも気に入らない．欲をいえば，486DX（20MHz）は欲しい．SXならば20MHzは最低ラインではないかと思う．この環境がなければBorland C++ V.3でWindowsのDOS/Vマシンに買い換えようかと思う，今日この頃です．（荒尾英一）

——いまFAユーザーの希望の星は，オーバードライブプロセッサです．でも安くはないですね．

★6月号のp143の答え

レタックス（電子郵便）のまちがいでした．僕がパソコン通信をやらないので，つい，不注意にも電子メールなんて言葉を使っていたのです．ゴメンなさい．えっ，結果ですか．どうもパソコン業界が混乱しているので，もう1年，情報工学の勉強するのを延期しました．（チャンドラー博士）

——「電子メール」で合格通知を受け取るはずだったチャンドラー博士さん，残念でしたね．もう1年がんばってください．

★ノートパソコンじゃ大きすぎる

電子手帳の話題が最近少なくなりましたが，電子手帳とほぼ同サイズで，表計算（1-2-3，マルチプランのような）が使用でき，漢字が使用できるものの出現をまっているんですがねェー．ノートパソコンは，やっぱり大きすぎます．セカンドバッグの中に入らないとねェー．（SG. my）

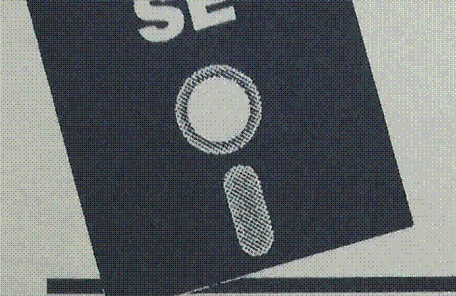
——英語ならそういう機械もあるんですがねェー．いちばん軽いノートパソコンでも1キロ以上ありますから，もって歩くのはけっこうつらいものです．

★複雑

既に，80286を載せた我愛機PC-9801UXも，中途半端な古いマシンとなってしまっている．しかし，買い換えには大きな抵抗が有る．何と言っても実際に自分が利用する範囲内では，まだまだ高機能マシンと言えるのだから．新製品を試す上では，確かに役不足なのだが……．複雑な心境である．（長谷川隆嗣）

——パソコンを使っているとこうしたジレンマに襲われます．対応は？　そのまま使い続ける，タイミングを見てグレードアップする，常に最新のものを使う，のいずれかだと思いますが．

（敬称略）



付録ディスクのご案内

このフロッピーディスクに含まれる実行ファイル，ソースリストおよびテキストファイルは「月刊Programmer's Page」1992年8月号に掲載されたものです．

付録のフロッピーディスクは，5"2HD（8セクタ／トラックMS-DOS）でフォーマットされています．3.5"や2DDなどへのコンバートサービスは行っておりません．なお，オンラインサービスSE-NET（03-5467-0376）でも，同じ内容のファイルを用意しています．

フロッピーディスクの内容は次の通りです．本誌およびディスクに含まれるドキュメントファイルをよく読んでからご利用ください．

■ディスクのおもな内容

●付録ディスクについての説明
ディレクトリ名：ルート
収録ファイル名：
README.BAT（ディスク内容の説明を読むためのバッチファイル，PC-9801専用）
README.DOC（ディスク内容の説明，MS-DOSテキストファイル）
ONLINE.BAT（オンラインソフトの説明を読むためのバッチファイル，PC-9801専用）
ONLINE.DOC（収録されているオンラインソフトの簡単な説明）
V.EXE（ミニページャー，PC-9801専用）
BOOK.TXT（翔泳社書籍目録）

●特集「進化するWindowsの開発環境」
①「Visual Basicによる実践的プログラム開発」（p.3）
ディレクトリ名：TOKU
収録ファイル名：
VB.LZH
MODULE.LZH
VSSAVE.LZH
②「Microsoft C/C++ Development System for Windows Version 7.0」（p.30）
ディレクトリ名：TOKU
収録ファイル名：C7.LZH
③「Borland C++ & Application Frameworks 3.0」（p.53）
ディレクトリ名：TOKU
収録ファイル名：BC.LZH

●プログラム「遊」工房（p.78）
「DROP/PICKUP/TRASH」ファイル削除／復活プログラム
ディレクトリ名：PLAY
収録ファイル名：PLAY.LZH

●実践C塾　アルゴリズム篇（p.83）
応用・その2
ディレクトリ名：JUKU
収録ファイル名：JUKU.LZH

●98活用アセンブラ技法　キーボード編（p.95）
キーボードドライバの作成
ディレクトリ名：98ASM
収録ファイル名：98ASM.LZH

●WordBASICを使おう（p.109）
ディレクトリ名：WORD
収録ファイル名：WORD.LZH

●綴じ込み付録
ディレクトリ名：DATA
収録ファイル名：MFCLIB.TXT（MS-DOSテキストファイル）

●オンラインソフト採集帳（p.144）
今月号で紹介したmmmオフラインシミュレータ「Hyper mmm」，mmm系WTERM用オートパイロットマクロ「wtmmm」および，斜め読み支援ツール「Askance」と，1つまたは複数のファイルの圧縮および「.LZH」という拡張子をもった圧縮ファイルの復元をする高圧縮復元書庫管理ソフト「LHA」が収録されています．なおそれぞれのファイルはLHAによって，LHAは自己復元型で，それぞれ圧縮されています．復元方法についてはONLINE.DOCを参照してください．
ディレクトリ名：ONLINE
収録ファイル名：
HMMM03ZD.LZH（Hyper mmm，MS-DOS汎用）
WTMMM16D.LZH（wtmmm，WTERM用マクロ）
ASKANT12.LZH（Askance，PC-9801対応）
LHA213.EXE（LHA v.2.13，MS-DOS汎用）

●クイズ・ランダムスクリーン（p.150）
クイズ解読ツール「PUZLE」
本誌掲載のクイズ・ランダムスクリーンの画像をディスプレイ上に再現し，マウスを使って並べ換えることにより，解答を得ることができるユーティリティです．PC-9801シリーズ専用です，詳しい使い方はPUZLE.DOCを参照してください．
ディレクトリ名：OMAKE
収録ファイル名：
PUZLE.EXE（解読ツール，PC-9801専用）
PUZLE.DOC（PUZLEの説明）
CONFIG.SYS（PUZLE用CONFIG.SYS）
AUTOEXEC.BAT（PUZLE用AUTOEXEC.BAT）
RNDSCREN.LZH（画像データ）
HIKOKI.LZH（7月号の解答データ，Windows 3.0の壁紙）

●そのほか
光システム（株）の協力によりリモート操作システム「バックマン」（PC-9801対応）を収録しています．
ディレクトリ名：BACKMAN
収録ファイル名：
BACKMAN.LZH（バックマン試用版）

※使用上の注意

本誌に掲載されたプログラムおよびデータを利用するには，MS-DOSの基礎知識が必要です．このディスクには，MS-DOSのシステムは入っていません．別途，MS-DOSが必要となります．また，ソースファイルのコンパイルなど，別途コンパイラやユーティリティが必要な場合があります．

■圧縮ファイルについて

ディスクスペースを有効に使うために，多くのファイルは圧縮した上で収録しています．「.LZH」という拡張子のついたファイルは，高圧縮書庫管理ユーティリティ「LHA」によって圧縮されたファイルです．これらのファイルを利用するには，まずLHAで復元する必要があります．LHAの利用方法については，ONLINE.DOCをご覧ください．

バーチャルスペースガイド

P子　某ソフトハウスで企画営業を担当している．26歳．
C平　某メーカーのシステムソフトウェア開発部門に所属．25歳．

C平　SF映画のリストを見ていると意外にタイムトラベルものって少ないんだよね．あっても荒唐無稽な超未来の話になったり，変則的な歴史映画であったり……．荒唐無稽でも，とにかく面白くできていたのは「バック・トゥ・ザ・フューチャー」だけど．
P子　物理学の理論だと時間旅行は不可能とされているわよね．
C平　未来へなら理論的には可能なんだ．光速に近い速さの宇宙船に乗って旅をすると，宇宙船内は時間の進み方が遅くなって，数年間して戻ってくると，そこはもう数百年も後の世界になっているということなんだけど……．
P子　でも，そういうのは旅行じゃないよ．戻れないんだから．
C平　歳をとらないという人工冬眠によって未来社会で目覚めるというのと変わりない．過去を再現しようとすると，オープンセットを組んだり，アニメにしたりとか，バーチャルなものにしかならない．過去を不思議にリアリティを表現できるのがアニメなんだね．「となりのトトロ」を見てそう思っちゃった．
P子　私もずいぶん前に見たよ．子供の頃の気持ちとかはよく表現されていたと思う．でも，C平と違って私は都会の人だから，あの田園風景にノスタルジアを感じたりはしないのさ．
C平　地方のほうが変化が激しいんだよ．田んぼの畦道がいきなりバイパスになったりして，田んぼのままだったら別に何も感じないんだけど……．
P子　で，トトロの田園風景でしんみりしてしまうわけね．いい子，いい子．

■となりのトトロ

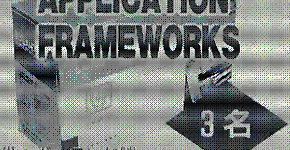
(1988年)
監督・原作・脚本：宮崎駿
9,260円（税別/LD）徳間書店・徳間コミュニケーションズ

昭和30年代初めを舞台にしたファンタジー．おとうさんとともに郊外の田園地帯に引っ越してきたサツキとメイ，おかあさんは病気で入院している．2人は，古い家や森にいる不思議な生きものと出会う．新しい生活の喜びや不安を子どもたちと不思議な生きものの結びつきを中心に描く．



読者プレゼント

1 BORLAND C++ APPLICATION FRAMEWORKS　3名



提供：ボーランド(株)

2 LSI C-86*1　5名



提供：エル・エス・アイ・ジャパン(株)

3 DiskXⅡ　3名



提供：エー・アイ・ソフト(株)

4 フリーソフトウェア WTERM＋テキスト　3名



翔泳社新刊

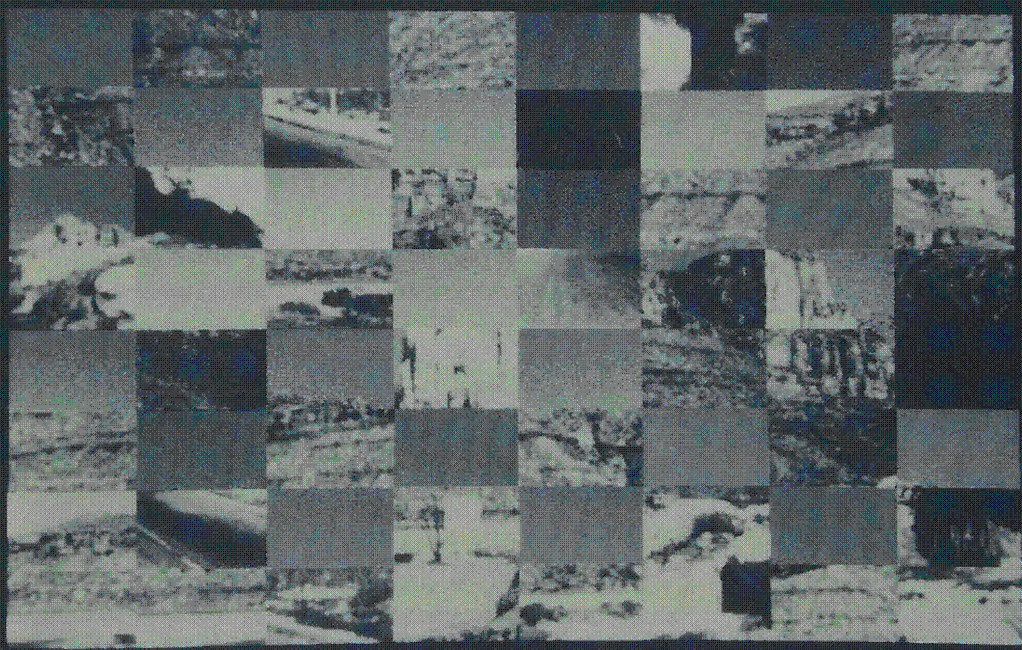
応募は本誌に添付のアンケートはがきをご利用ください．(1992年8月17日消印有効)．なお，応募者多数の場合は，抽選になります．当選者発表は発送をもって代えさせていただきます．

*1：5インチのみ．

クイズ・ランダムスクリーン

ある1枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：○○○○マウンテン

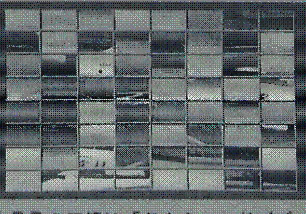


正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．
応募方法：本誌に添付のアンケートはがきをご利用ください．
締め切り：1992年8月17日（消印有効）
なお，当選者の発表は発送をもって代えさせていただきます．

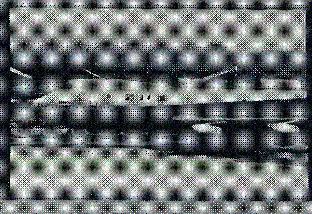
＊画像データは，付録ディスクやBBSでもサービスしています．

7月号の問題と正解

【問題】



【正解：飛行機】



＊今月号の正解は「ドクター・ドブズ・ジャーナル/日本版創刊号」に掲載します．

SE-NETからのお知らせ

SE-NETでは開設以来，オンラインサービスを行ってきましたが，回線混雑のため非常に接続しにくい等の問題が出てまいりましたので，GUESTによるダウンロードは中止させていただきます．今後は読者のみなさまと編集部とのコミュニケーションの場としてご利用ください．

●BBSへの接続

SE-NET　03-5467-0376

通信速度	300/1200/2400bps
接続プロトコル	CCITT V.22bis/V.22/V.21
エラー訂正プロトコル	CCITT V.42，MNPクラス5
漢字コード	シフトJIS

＊移転のため電話番号が変わっておりますので，ご注意ください．

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ バックナンバーのご案内

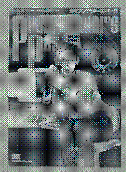
1991年1月号「特集・プログラマの仕事場」・2月号「特集・Windowsの時代である!」・3月号「特集・この言語を使いなさい!!」・4月号「特集・C読本」・5月号「基礎力としてのアセンブラ」・7月号「特集・最新デバイスドライバ研究」・1992年5月号「特集・Cの学び方・教え方」は品切れです。

■定価・各980円

1991年6月号 5インチFD付き

特集・研究「プログラミング統合環境」

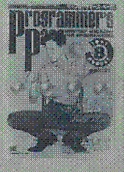
今流行のエディタ，コンパイラ，リンカ，デバッガを統合した開発環境について，その歴史的系譜から最新のシステムまで徹底的に検証する。



1991年8月号 5インチFD付き

特集・はじめてのオブジェクト指向プログラミング

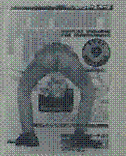
"オブジェクト指向"の歴史的な発展の過程から基礎概念，分析・設計の方法までを網羅的に解説する。



1991年9月号 5インチFD付き

特集・グラフィックスプログラミングの世界

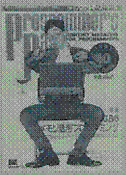
基礎からフレームバッファの活用，レイトレーシング，Windows，IBM PCのグラフィックスなど，その技法を徹底的に解説する。



1991年10月号 5インチFD付き

特集・286/386メモリ活用プログラミング

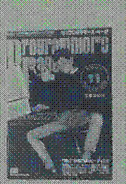
バンク式メモリ，EMS，XMSからプロテクトモードまで，640KBの壁を破るための数多くのプログラミングテクニックを解説する。



1991年11月号 5インチFD付き

特集・Windowsプログラムの作り方

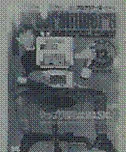
正統派MS-C+SDKをはじめ，VisualBasic，BC++，TPWなどの新しい開発環境，ツールによるWindowsプログラムの作り方を探る。



1991年12月号 5インチFD付き

特集・やっぱり僕らはBASIC!

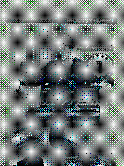
構造化プログラミング，マルチランゲージに対応した強力なプログラミング言語として生まれ変わった現在の進んだBASIC，MS-BASIC 7.1，TrueBASICからVisualBasicまでを紹介する。



1992年1月号 5インチFD付き

特集・プログラミングツール大集合

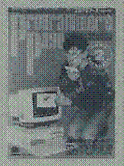
プログラム開発，メンテナンスに欠くことのできないプログラミングツール，その利用法から，使いこなしのコツまでを紹介する。



1992年2月号 5インチFD付き

特集・通信ソフトの作り方

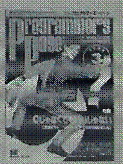
RS-232Cを使用した通信の概要を解説し，ライブラリ，ライブラリを用いた通信用マクロ言語を作成，BBSなどにアクセス可能な通信プログラムを完成する。



1992年3月号 5インチFD付き

特集・Cじゃなくてもいいじゃない

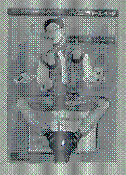
C言語の実体をとらえなおし，Pascalや MODULA-2など，ほかのプログラミング言語と比較することでプログラミング言語の能力と性能を解説する。



1992年4月号 5インチFD付き

特集・新しいDOS環境を探検する

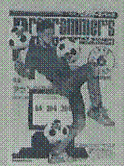
これまでのDOS環境に変化をもたらしたMS-DOS5.0，DOS/V上でのプログラミングを探る，バージョンアップしたMASM6.0もあわせて紹介。



1992年6月号 5インチFD付き

特集・アセンブラの作り方

MS-DOSと8086系CPUを題材に，具体的にC言語によるソースコードを示しながら，486までの命令を処理できる8086系簡易アセンブラの作成方法を紹介する。



1992年7月号 5インチFD付き

特集・過激なプログラミング入門

プログラミングの最も重要な過程であるプログラム設計と，CPUのマシン語命令がどのような意図に基づいて設計されるかを学ぶという両極からのプログラミング入門を試みる。



「月刊プログラマーズ・ページ」のバックナンバーは，ご注文いただければ，全国どこの書店でもお求めになれます(書店に届くまで1か月ほどかかります)。小社に直接ご注文なさるときは，何月号かを明記の上，現金書留にて雑誌価格＋送料を右記までお送りください。送料は1部81円です。2冊以上の場合は，号数・冊数により送料が異なりますので，小社までお問い合わせください。

〒106 東京都港区西麻布2-22-1
スリーアップ西麻布ビル
(株)翔泳社
マーケティング部PP係宛
TEL 03-5467-0361

新創刊

Dr. Dobb's
JOURNAL JAPAN

月刊ドクター・ドブス・ジャーナル／日本版
創刊号（9月18日発売）

特集 1
プロテクトモードプログラミング
特集 2
新しいWindowsプログラミング

特集 1では，WindowsにおけるDPMI，リアル・プロテクトモードコードのミックス，UNIXプログラムのDOSへの移植法をとりあげ，特集 2では，Windows 3.1, NTにおけるWindowsプログラミングの実際をツール・環境からテクニックまで紹介する．

月刊プログラマーズ・ページ　1992年8月号
1992年8月1日発行
発行所　株式会社翔泳社
東京都港区西麻布2-22-1スリーアップ西麻布ビル
TEL 03-5467-0361　〒106
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋芳山充弘
表紙デザイン　SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋種村嘉彦
協力　根本裕三郎＋(株)ドモン・マインズ＋保坂庸介＋小林弘人(図版デザイン)
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　株式会社サンニチ印刷
定価　980円（本体951円，税29円）

EDITOR'S PAGE

●昨年１月号での創刊以来，20号にわたって発行してきたプログラマーズ・ページは，本号をもって区切りをつけ，新装開店することになった．現在のプログラマーズ・ページを支持してくださる読者も少なくないことを思うと心苦しいが，より幅広い読者に求められる情報をコンスタントに提供していくことを考えての結論である．誌名は変わっても同じフィールドを対象にした雑誌であることには変わりない．どうか新創刊後もご愛読いただきたい．(TA)

●ようやくプログラマーズ・ページの認知度も上がってきたかなと思っていたら，誌面刷新・新創刊である．思い起こせば１年と８か月，試行錯誤の連続だった．ずいぶんと突拍子もない特集やら，まとまりのつかない散漫なものやら，人目を惹かない地味なものやらを作ってきたが，それもこれも今となっては懐かしい思い出である．こうして，また！　１つの雑誌が消えていくのだ（新しく生まれ変わるけど……），朝駆け・夜討ちの原稿催促ともしばしの別れ(笑)，最後にもう一発原稿催促の電話を入れるかな……（^^）．(嘉平)

●ながいようでみじかい20カ月でした．もう少し時間があれば，ああもしたい，こうもしたいと思いつつも，自分の非力さに腹が立ち，歯がゆい20カ月でもありました．とにもかくにも，皆さんお世話になりました．また，DDJJでお会いしましょうね．(ま)

●RAM, MO……などなどの衝動買い，ついに最高額を更新してしまった．事件は６月上旬のある晴れた日曜日の午後に起こった．事前に，「その場で買ったりして」「いくらなんでもそんな高額なもの，お金がないよ」といった会話を某氏と交わした記憶がある．…でも某氏の予想は当たっていた．買ったのは青いユーノスロードスター，でも運転者は火の車…(笑)(←笑い事ちゃないってば)，ガソリンがないと車は動かないと思うんだけど，そんなお金はどこにあるんだろう．(Macintosh Ⅱcxも最近わらしべたyoshi)

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊**プログラマーズ・ページ**
READER'S SURVEY CARD

愛読者カード

本誌の号数　　　年　　　月号

本誌を知った理由
- □1.書店店頭　□2.他社出版物（　　　）
- □3.小社出版物　□4.DM　□5.口コミ

本誌を購入した書店　　　市（区）　　　書店

パソコンの所有形態（機種名）
- □1.個人所有　（　　　）
- □2.職場にある（　　　）
- □3.持っていない

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事（いくつでも）

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト3

パソコン雑誌	①	②	③
ソフトウェア	①	②	③
パソコン本体	①	②	③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください（「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください）。

月刊**プログラマーズ・ページ** 広告資料請求カード

8	会社（学校）名		9	所属（学部）		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業　④建設業　⑤通信・運輸　⑥システム・ソフト　⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ（ハード）技術者　②コンピュータ（ソフト）技術者　③エレクトロニクス技術者　④プログラマ　⑤その他の技術者　⑥専門職　⑦事務・営業職　⑧経営管理職　⑨商工サービス　⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　年　　月号

広告資料請求No. / 希望項目												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

月刊ドクター・ドブズ・ジャーナル/日本版　定期購読のご案内

綴じ込みの振替用紙を使って、定期購読をお申し込みになれます。
1年間12冊=11,760円(本体11,417円/税343円)
郵送料サービス

郵便はがき

料金受取人払
麻布局承認
905

差出有効期間
平成6年4月
30日まで

切手不要

106-00

東京都港区西麻布2-22-1
スリーアップ西麻布ビル

株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ READER'S SURVEY CARD 愛読者カード

氏名	性別 □1男 □2女	生年月日 19　年　月　日
住所 〒		電話 (　　) －

職業
□1ソフト技術者 □2ハード技術者 □3技術系会社員
□4事務系会社員 □5営業系会社員 □6教師
□7一般公務員 □8経営管理職 □9理系学生
□10文系学生 □11その他(　　　)

ユーザー区分
□1 プロ
□2 マニア
□3 一般ユーザー

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払
麻布局承認
906

差出有効期間
平成6年4月
30日まで

切手不要

106-00

東京都港区西麻布2-22-1
スリーアップ西麻布ビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

＊当社で記入

1	SE	2	受付No.

月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓　　名	5	生年月日	19　年　月　日
6	自宅住所	〒　電話　(　　)			
7	会社住所	〒　電話　(　　)			

資料送付希望先の番号に○印をつけてください。

報を

ジが

払込通知票

口座番号 東京 4-154493

加入者名 株式会社 翔泳社

金額 11760

払込票

口座番号 東京 4-154493

加入者名 株式会社 翔泳社

金額 11760

Dobb's

NAL JAPAN

ウェア開発技術を提供する充実した情報
誌づくりをめざします。引き続きご愛読
くださいますようお願いいたします。

翔泳社

SE
SHOEISHA

定期購読のご案内

- ■定期購読は、1年間12冊と限定させていただきます。
- ■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。
- ■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。
- ■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。
- ■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

この欄は、加入者あての通信にお使いください。

ドクター・ドブズ・ジャーナル/日本版　定期購読直接申し込み書

Dr.Dobb's JOURNAL JAPAN

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵　政　省）

郵便はがき

1 0 6 - 0 0

東京都港区西麻布2-22-1
スリーアップ西麻布ビル

株式会社 翔泳社

月刊プログラマーズ・ページ
広告資料請求係行

2	受付No.	

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名	5	生年月日	19　年　月　日

□7 一般事務員　□8経営管理職　□9理系学生　□10文系学生　□11その他(　　　)　□ 一般ユーザー

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　電話（　　）	資料送付希望先の番号に○印をつけてください。
7	会社住所	〒　電話（　　）	

月刊ドクター
綴じ込み
1

より新しく、より深い技術情報をお届けするために月刊プログラマーズ・ページが生まれかわります。

月刊プログラマーズ・ページは、1991年1月号での創刊以来、多くの読者の支持を得て本号で通巻20号を数えました。パソコンプログラミングという変動の大きな分野において、本誌を支えていただいた読者のみなさまに深く感謝いたします。
このたび小社では、米国M&T Publishing社との間で、同社の発行するDr. Dobb's JOURNAL誌に関する提携契約を締結いたしました。これを機に月刊プログラマーズ・ページとしての歴史に区切りをつけ、より新しく深い技術情報を提供する月刊誌「ドクター・ドブス・ジャーナル／日本版(Dr. Dobb's JOURNAL JAPAN)」として新しくスタートすることになりました。プログラマーズ・ページと同様に、パーソナルコンピュータを中心にソフトウェア開発技術を提供する充実した情報誌づくりをめざします。引き続きご愛読くださいますようお願いいたします。

翔泳社

SE
SHOEISHA

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

人材募集広告企画

Microsoft®

マイクロソフト株式会社

ちょっと気になるこの社会

1986年の設立以来5年間で480%という驚異的な成長を遂げてきたマイクロソフト株式会社。
その歩みは日本のパーソナルコンピュータの発展とともにあったと言ってもいいだろう。
その研究開発部門の第一線での活躍ぶりを拝見しに、新宿の本社オフィスを訪ね、第一研究開発部アプリケーション1課主事の中島俊朗氏、同課新人の石坂直樹氏、そして人材開発部課長の佐藤和男氏からお話をうかがった。

ソフトウェア開発の頂点をめざして

——なぜコンピュータの仕事に就こうと思ったのか、まず聞かせてください。

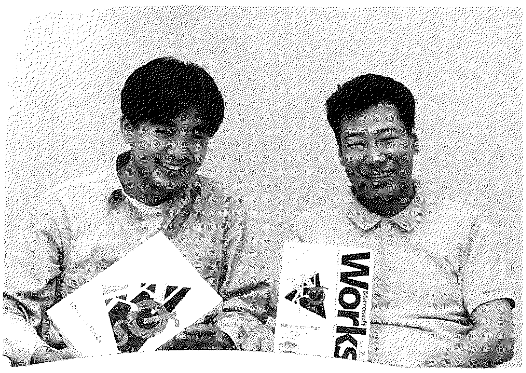
石坂　私はこの春新卒で入社したばかりなんですが、大学の社会学部でメディアやコミュニケーションについて学ぶ中で、これからのメディアはコンピュータなしでは語れないと考えまして、それなら一番中心にあるコンピュータの世界をめざそうと思ったわけです。

中島　私がこの業界に関わるようになったのは8年くらい前でした。それ以前はハードウェア関係の仕事をしていたんですが、ソフトウェアがこれからは重要じゃないかと考えて、パソコンソフトの会社に移ったのがはじまりですね。

——その中で、マイクロソフトへの入社を決めた理由は。

石坂　やはり、コンピュータのソフトウェアの中で優れた技術を自社で持っているという点ですね。

中島　私は3年前に当社に転職したんですが、パソコンのソフトをやるなら、そのトップクラスであるマイクロソフトしかない、という気持ちでした。



中島俊朗氏　33歳
某ソフトウェア会社を経て
1989年6月
マイクロソフトへ入社

※写真右

石坂直樹氏　22歳
法政大学　社会学部
社会学科卒
1992年4月入社

※写真左

『Works』開発の第一線で

——入社前と入社後とでは、会社に対する見方に変化がありましたか。

石坂　入社前は外資系会社という意識でしたが、実際に入社してみると外資系だなという感じではなくて、日本法人のやり方が会社の中でうまく生かされているなと感じました。

中島　僕はプログラマーとして入ったので、マイクロソフトならきっとすごく整然としてきれいなコードを書くんじゃないかと興味をもっていたんですよ。ところが実際に見てみると、ソースがきたないんです(笑)。しかしよく読んでいくと、むだなものがなく、非常にオプティマイズされたコードで、なるほどこれ以外に書きようがない、最高のコードで書かれているんだな、ということがわかってきて、改めて感心したということがありましたね。

——現在はお二人とも『Works』のチームにいらっしゃるということですが。

中島　ええ、僕はWorks担当で、基本的には開発のプログラミングの方の担当なんですが、実際には全体をコーディネイトしたりする仕事の方がかなり多くなってきました。

石坂　4月に入社後、1ヶ月間は仮配属で営業を経験して、5月から研究開発部門であるR&Dに本配属になりました。現在はWorksの中でテスティン

グ、品質保証のさまざまなテストを行う仕事を担当しています。

——職場の先輩たちの印象は。

石坂　プロ意識の高い方が集まっているなという印象です。中島さんを始め皆さん、自信に裏付けられた冷静さや落ち着きを持っておられると思います。

中島　そんなに持ち上げなくても(笑)。まあ、この会社のいいところは人の短所を見つけあうのではなく、長所、つまり自分の得意な分野を互いに出し合って、いいものを作り上げていこうというアグレッシブな姿勢でしょうね。石坂君もこれから、自分でやりたいことをどんどん見つけて、もっと自分を押し出していくといいと思う。

石坂　まだ新人なので、様子を見てるんですよ(笑)。これから少しずつ出していこうかと。

佐藤　たとえて言えば、当社はサラブレッドの集団であることを望んでいないんです。むしろ動物園のような場所であればいい。熊もいればキリンもいる、ライオンもいればゾウもいるという集団ですね。それくらい一人一人が独自の発想を持ち、一つの目的に向かっていろんな意見を出し合いながら、いいものをつくっていってほしい。

苦労した仕事ほど、楽しみも大きい

——仕事の上での苦労や楽しみというと、どんなことが頭に浮かびますか。

中島　苦労した仕事ほど、また楽しかったと思いますね。今の仕事の中では、Worksの2.1を出す直前の2、3ヶ月がいちばん苦しかった時期で、忙しいというだけではなく、精神的にもぎりぎりでやっていた時期だったなと思います。米国の本社へ9ヶ月ほど行ってWorksを作り、それを日本に持ち帰って仕上げるまでですね。プログラムそのものより、それを商品として仕上げる段階がいちばん苦労します。いろいろな人との共同作業ですから、話し合い、調整しながら、さまざまな問題をクリアしていくプロセスが大変でした。また、その苦労があったからこそ、印象に残る楽しい仕事になったんでしょうね。

石坂　私は研修期間中にプロジェクトの会議に参加させてもらったのがいい経験になりました。英語だけの会議に出たときはかなりプレッシャーも感じましたし、週や月単位で提出するレポートをすべて英語で書くというのも大変でした。それが辛くもあり面白くもあったということですが、これからの覚悟を固めるためにはいい機会だったんじゃないかと思います(笑)。

佐藤　新人でも重要なミーティングに参加させてしまうのがマイクロソフトらしいところでしょうね。若い社員にも重要な仕事を任せる、チャンスを与えるという考え方ですから。

——英語環境の話が出ましたが、英語能力はかなり必要とされるんですか。

佐藤　日常使う電子メールから、会議、出張まで英語力が必要とされる場面は多いですね。しかし入社当初はできなくても、やる気があれば飛躍的に伸びていきますよ。会社としても英語学習の環境をきちんと整えていますし。

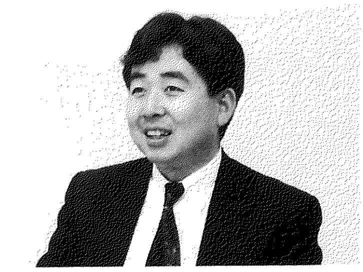
中島　僕も入社当時は語学がだめだったんですが、自分が必要としていましたから、完全な英語ではありませんが、意思を伝えることはできるようになりました。そういう気持ちさえあれば身につくものだと思います。

どこまでもチャレンジの精神で

——今後、この仕事を続けていく上で、大切にしていきたいことは何ですか。

石坂　ユーザーあるいはこれからユーザーになる人たちを常に念頭において、十歩先を見ながら、二歩か三歩か先をゆく商品を作ることを考えなければいけないと思っています。

佐藤　将来を考えると、MS-DOSがパソコンだけでなく、テレビやカメラなどいろいろな機器の中に入っていく可能性も大いにあるわけで、当社の仕事はパソコンのソフトウェアからさらに広がっていくでしょう。石坂君たち若い人が活躍する場が今後もたくさん待ち構えていると思いますよ。



人材開発部　課長
佐藤和男氏

中島　技術革新がたいへん速い世界ですので、いままでの技術がすぐ陳腐化する可能性も大きいですね。そういう意味では常に敏感になっていなくてはいけないし、向上心といいますか、自分の世界はここまでだという壁を作らないで、自分の可能性に自信を持って、チャレンジしていくことが必要なんじゃないかと思います。

佐藤　マイクロソフトという会社は、会社にぶら下がって仕事をやっていこうという人のためにあるのではなく、マイクロソフトに入って自分の可能性を試してみたいと思っている人に開かれた会社だということですね。当然、技術力も必要ですし、チームワークも大切、また英語力も必要になってきますが、最初からそれらすべてを兼ね備えた人間はいないんで、だからこそ、チャレンジ精神をもって、これをやるんだという目的と意欲を持った人たちに集まってきてほしいですね。

アプリケーション開発に関わる苦労、楽しみ、夢を語る中島氏と石坂氏。会社組織としてのマイクロソフト社を採用担当の立場から語る佐藤氏。会談は、終始自社に対する自信と信頼に満ち溢れた雰囲気に包まれていた。マイクロソフト社は決して受身ではなく、積極的に自分の実力を発揮できる申し分のない環境である。マイクロソフトというフィールドであなたも自分の可能性と創造性を試してみてはいかがだろう。

募集職種

ソフトウェアデザインエンジニア
●仕事内容
システムソフトウェア、MS-DOSまたはWinbows用アプリケーションソフトウェア及び言語ソフトウェアの開発。
●応募資格
C、C++、及び8086系アセンブラでのプログラミング経験がある方。
ウインドウシステム(Windows等)の開発に経験又は知織のある方。
Windows用アプリケーションの開発経験がある方。

プログラムマネージャ
●仕事内容
製品の工程管理や仕様の決定、開発プロジェクトの管理、海外の開発プロジェクトとの調整等。
●応募資格
ソフトウェアの企画・デザインに知識のある方。
ソフトウェア開発プロジェクトのコーディネーションに知識のある方。

データベース技術者
●仕事内容
Windows用データベース管理ソフトの研究、開発、及び品質管理。
●応募資格
Windows用アプリケーション及びデータベース管理ソフトの開発経験がある方。
データベース管理の理論について知識のある方。

ソフトウェアテスティングエンジニア
●仕事内容
ソフトウェア開発プロジェクトにおける品質管理全般、テストプラン立案並びにスクリプトの開発、スケジュール管理、出荷基準のプランニング。
●応募資格
MS-DOS及びWindows環境の基礎知識のある方。
マクロ言語及びC言語を主体としたソフトウェア開発の経験がある方。
テスティングの自動化経験がある方。

ユーザーエデュケーション
●仕事内容
ソフトウェアのペーパーマニュアル、オンラインマニュアル、学習ソフトの企画、編集、原稿作成、工程管理等。オンラインマニュアル、学習ソフトに関するプログラミング。
●応募資格
ソフトウェアのマニュアル、オンラインマニュアル制作の経験がある方。
一般書籍などの関係職種でのマニュアル制作の経験がある方。
オンラインマニュアルに関するプログラミングの経験がある方。

目的を持つ人のための開かれた環境

Microsoft®

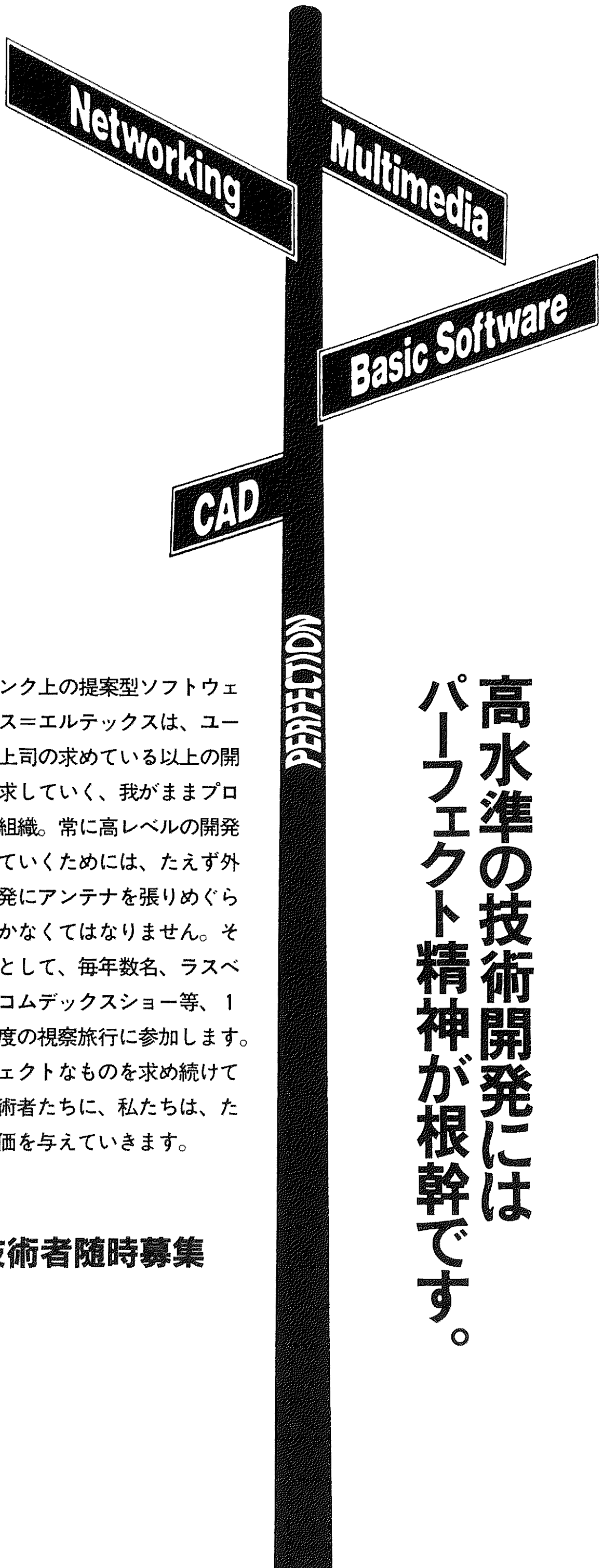
募集要項

年　　齢■22歳～28歳位まで。(来春新卒者可)
給　　与■経験・年齢等を考慮の上、当社規定により優遇。
勤 務 地■東京都新宿区(本社)
待　　遇■昇給年1回・賞与年2回
勤務時間■9:00～17:30
フレックスタイム制(コアタイム11:00～15:00)
休　　日■完全週休二日制(土・日)、祝日
応募方法■希望職種を明記の上、履歴書(写真貼付)と職務経歴書をご郵送下さい。
書類選考の上、面接日を連絡致します。
＊応募秘密厳守　＊応募書類返却不可
設　　立■1986年2月17日
資 本 金■1億円
売 上 高■90億円(1991年6月期)
代表取締役社長■成毛　真
従業員数■330名

事業内容■パソコン用ソフトウェアの開発及び販売
米国本社■マイクロソフト・コーポレーション(米国ワシントン州レッドモンド)
連 絡 先■マイクロソフト株式会社
(芝オフィス)　人材開発部＜S係＞
〒105 東京都港区芝1-11-11
住友不動産芝ビル
TEL 03(5484)1420

新宿オフィス(勤務地)
〒160 新宿区西新宿7-5-25 Kビルディング
Tel.03-3363-1200(代表)





高水準の技術開発には パーフェクト精神が根幹です。

ワンランク上の提案型ソフトウェアハウス＝エルテックスは、ユーザーや上司の求めている以上の開発を追求していく、我がままプロ集団の組織。常に高レベルの開発を行っていくためには、たえず外部の開発にアンテナを張りめぐらしておかなくてはなりません。その一環として、毎年数名、ラスベガスのコムデックスショー等、1週間程度の視察旅行に参加します。パーフェクトなものを求め続けていく技術者たちに、私たちは、たえず評価を与えていきます。

技術者随時募集

会社概要

ELTEX

設　立	昭和60年12月14日
資本金	1000万円
	(1)ネットワークの端末システムの開発 (2)ＣＤ－ＲＯＭ、ＬＤなど新しいメディアに対応したシステムの開発 (3)ＯＣＤ、ＣＡＤなどのソフトウェアプロダクトの開発、販売 (4)ビジネスパッケージの開発、販売
特　徴	大手クライアント向けにネットワークやマルチメディアなどのシステム開発を幅広く対応している。また、手書文字読み取りシステム「読造」*1、水道工事店向け「配管ＣＡＤ」*1などの自社ソフトウェア製品の開発販売や、電子回路ＣＡＤ「Ｔａｎｇｏ」シリーズ*2の日本語化など積極的な事業展開をしている。 *1　当社で開発し三菱系ディラーで販売しているプロダクト *2　米アクセルテクノロジ社の製品をＰＣ9801への移植と日本語化したプロダクト、㈱ソーテック工人舎事業部で販売している。

募集要項

職　種	①システムエンジニア　②プログラマー
資　格	専門卒以上20～36歳の方　※Ｃ言語・アセンブラ言語経験者大歓迎　※未経験者でも好奇心・向学心の旺盛な方は歓迎します。
給　与	17万円（専門卒初任給）～46万円（36歳課長、諸手当含む） 24歳例：23万5000円（情報処理1種手当・住宅手当含む） 30歳例：31万6000円（中級SE、情報処理1種・住宅手当含む）
手　当	時間外手当、役職手当（5000円～６万円）、住宅手当（１万5000円）、資格手当（２種：5000～特種：２万円）、通勤手当
賞　与	賞与年２回（平成１年度実績年間５ヵ月）、決算賞与年１回（平成１年度実績１ヵ月）
勤務地	本社（横浜駅近く） ◇派遣・出向はありません。
勤務時間	フレックスタイム制（コアタイム10:00～16:00）試用期間中は標準時間9:00～17:45
休日休暇	完全週休２日制（土・日）、祝日、年３回連休（GW・夏期・年末年始）、有給・特別・休暇
福利厚生	各種社会保険完備、退職金制度、財形貯蓄、社員旅行、海外研修、定期健康診断、社員寮、スポーツクラブ法人会員
応募方法	電話連絡の上、履歴書（写真貼付）、職務経歴書を持参し来社ください。郵送も受け付けます。
連絡先	〒220 横浜市西区南幸二丁目20番５号　東伸24ビル５階　℡(045)322-9667　総務課

採用HOT・LINE

TEL.045-322-9667

株式会社エルテックス

Microsoft C/C++ Ver.7.0 ファウンデーションクラスライブラリ一覧

■CArchive (#include 〈afx.h〉) バッファリングされたバイナリストリーム操作をサポート

Public Members
- Construction/Destruction:CArchive, ˜CArchive, Close
- Basic Input/Output:Flush,operator >>,operator <<,Read,Write
- Status:GetFile,IsLoading,IsStoring

Protected Members
- Object Input/Output:ReadObject,WriteObject

■CArchiveException (#include 〈afx.h〉) バイナリファイルストリームの例外処理操作をサポート

Public Members
- Data Members:m_cause
- Construction/Destruction:CArchiveException

■CBitmap Windows GDIのビットマップコントロール機能をサポート

Public Members
- Construction/Destruction:CBitmap
- Initialization:LoadBitmap,LoadOEMBitmap,CreateBitmap,CreateBitmapIndirect,CreateCompatibleBitmap,CreateDiscardableBitmap
- Operations:FromHandle,SetBitmapBits,GetBitmapBits,SetBitmapDimension,GetBitmapDimension

■CBrush Windows GDIのブラシコントロール機能をサポート

Public Members
- Construction/Destruction:CBrush
- Initialization:CreateSolidBrush,CreateHatchBrush,CreateBrushIndirect,CreatePatternBrush,CreateDIBPatternBrush
- Operations:FromHandle

■CButton Windowsのボタンコントロール機能をサポート

Public Members
- Construction/Destruction:CButton
- Initialization:Create
- Operations:GetState,SetState,GetCheck,SetCheck,GetButtonStyle,SetButtonStyle

■CByteArray (#include 〈afxcoll.h〉) バイト配列をサポート

Public Members
- Construction/Destruction:CByteArray,˜CByteArray
- Bounds:GetSize,GetUpperBound,SetSize
- Operations:FreeExtra,RemoveAll
- Element Access:GetAt,SetAt,ElementAt
- Growing the Array:SetAtGrow,Add
- Insertion/Removal:InsertAt,RemoveAt
- Operators:operator []

■CClientDC CDCから派生するクライアントデバイスコンテキストクラスを定義

Public Members:Construction/Destruction:CClientDC,˜CClientDC

Protected Members:m_hWnd

■CComboBox Windowsのコンボボックス機能をサポート

Public Members
- Construction/Destruction:CComboBox
- Initialization:Create
- General Operations:GetCount,GetCurSel,SetCurSel,GetEditSel,LimitText,SetEditSel,GetItemData,SetItemData,GetLBText,GetLBTextLen,ShowDropDown,Clear,Copy,Cut,Paste
- String Operations:AddString,DeleteString,InsertString,ResetContent,Dir,FindString,SelectString

■CDC デバイスコンテキストオブジェクトクラスを定義

Public Members
- Construction/Destruction:CDC,˜CDC
- Initialization:CreateDC,CreateIC,CreateCompatibleDC,DeleteDC
- Device-Context Functions:GetDCOrg,SaveDC,RestoreDC,GetDeviceCaps
- Drawing-Tool Functions:GetBrushOrg,SetBrushOrg,EnumObjects
- Type - Safe Selection Helpers:SelectObject,SelectStockObject,Color and Color Palette Functions,GetNearestColor,SelectPalette,RealizePalette,UpdateColors
- Drawing - Attribute Functions:GetBkColor,SetBkColor,GetBkMode,SetBkMode,GetPolyFillMode,SetPolyFillMode,GetROP2,SetROP2,GetStretchBltMode,SetStretchBltMode,GetTextColor,SetTextColor
- Mapping Functions:GetMapMode,SetMapMode,GetViewportOrg,SetViewportOrg,OffsetViewportOrg,GetViewportExt,SetViewportExt,ScaleViewportExt,GetWindowOrg,SetWindowOrg,OffsetWindowOrg,GetWindowExt,SetWindowExt,ScaleWindowExt
- Coordinate Functions:DPtoLP,LPtoDP
- Region Functions:FillRgn,FrameRgn,InvertRgn,PaintRgn,Clipping Functions,GetClipBox,SelectClipRgn,ExcludeClipRect,ExcludeUpdateRgn,IntersectClipRect,OffsetClipRgn,PtVisible,RectVisible
- Line-Output Functions:GetCurrentPosition,MoveTo,LineTo,Arc,Polyline
- Simple Drawing Functions:FillRect,FrameRect,InvertRect,DrawIcon
- Ellipse and Polygon Functions:Chord,DrawFocusRect,Ellipse,Pie,Polygon,PolyPolygon,Rectangle,RoundRect
- Bitmap Functions:PatBlt,BitBlt,StretchBlt,GetPixel,SetPixel,FloodFill,ExtFloodFill
- Text Functions:TextOut,ExtTextOut,TabbedTextOut,DrawText,GetTextExtent,GetTabbedTextExtent,GrayString,GetTextAlign,SetTextAlign,GetTextFace,GetTextMetrics,SetTextJustification,GetTextCharacterExtra,SetTextCharacterExtra,Font Functions,GetCharWidth,SetMapperFlags,GetAspectRatioFilter
- Printer Escape Functions:Escape,StartDoc,StartPage,EndPage,SetAbortProc,AbortDoc,EndDoc
- Scrolling Functions:ScrollDC
- Metafile Functions:PlayMetaFile

■CDialog Windowsのダイアログボックス機能をサポート

Public Members
- Operations:MapDialogRect,IsDialogMessage,NextDlgCtrl,PrevDlgCtrl,GotoDlgCtrl,SetDefID,GetDefID,EndDialog
- Overridables:OnInitDialog,OnSetFont

Protected Members
- Construction/Destruction:CDialog
- Initialization:Create,CreateIndirect

■CDumpContext (#include 〈afx.h〉) ストリーム環境の診断出力をサポート

Public Members
- Construction/Destruction:CDumpContext
- Basic I/O:Flush,operator <<,HexDump
- Status:GetDepth,SetDepth

■CDWordArray (#include 〈afxcoll.h〉) 32ビットダブルワード配列をサポート

Public Members
- Construction/Destruction:CDWordArray,˜CDWordArray
- Bounds:GetSize,GetUpperBound,SetSize
- Operations:FreeExtra,RemoveAll
- Element Access:GetAt,SetAt,ElementAt
- Growing the Array:SetAtGrow,Add
- Insertion/Removal:InsertAt,RemoveAt
- Operators:operator []

■CEdit Windowsのエディットコントロール機能をサポート

Public Members
- Construction/Destruction:CEdit
- Initialization:Create
- Multiple - Line Operations:GetLineCount,GetHandle,SetHandle,FmtLines,LineIndex,SetRect,SetRectNP,SetTabStops
- General Operations:CanUndo,GetModify,SetModify,GetRect,GetSel,GetLine,EmptyUndoBuffer,LimitText,LineFromChar,LineLength,LineScroll,ReplaceSel,SetPasswordChar,SetSel,Undo,Clear,Copy,Cut,Paste

■CException (#include 〈afx.h〉) 全例外処理の基本クラスを定義

■CFile (#include 〈afx.h〉) ファイルクラスの基本クラスを定義

Public Members
- Data Members:m_hFile
- Construction/Destruction:CFile,˜CFile,Duplicate,Open,Close
- Input/Output:Read,Write,Flush
- Position:Seek,SeekToBegin,SeekToEnd,GetLength,SetLength
- Locking:LockRange,UnlockRange
- Status,GetPosition:GetStatus
- Static:Rename,Remove,GetStatus,SetStatus

■CFileException (#include 〈afx.h〉) ファイル関連の例外処理をサポート

Public Members
- Data Members:m_cause,m_lOsError
- Construction/Destruction:CFileException
- Code Conversion:OsErrorToException,ErrnoToException,Helper Functions,ThrowOsError,ThrowErrno

■CFont Windows GDIのフォント操作をサポート

Public Members
- Construction/Destruction:CFont
- Initialization:CreateFontIndirect,CreateFont
- Operations:FromHandle

■CFrameWnd SDIのポップアップウィンドウやオーバーラップウィンドウをサポート

Public Members
- Data Members:rectDefault
- Construction/Destruction:CFrameWnd,˜CFrameWnd
- Initialization:Create,LoadAccelTable
- Operations:GetChildFrame,GetParentFrame

Protected Members:Data Members:m_hAccelTable

■CGdiObject Windows GDIの基本クラスを定義

Public Members
- Data Members:m_hObject
- Construction/Destruction:CGdiObject,˜CGdiObject
- Operations:GetSafeHandle,FromHandle,Attach,Detach,DeleteObject,DeleteTempMap,GetObject,CreateStockObject,UnrealizeObject

■CListBox Windowsのリストボックス機能をサポート

Public Members
- Construction/Destruction:CListBox
- Initialization:Create
- General Operations:GetCount,GetHorizontalExtent,SetHorizontalExtent,GetTopIndex,SetTopIndex,GetItemData,SetItemData,GetItemRect,GetSel,GetText,GetTextLen,SetColumnWidth,SetTabStops
- Single Selection Operations:GetCurSel,SetCurSel
- Multiple Selection Operations:SetSel,GetSelCount,GetSelItems,SelItemRange
- String Operations:AddString,DeleteString,InsertString,ResetContent,Dir,FindString,SelectString

■CMapPtrToPtr (#include 〈afxcoll.h〉) voidポインタによって示されるvoidポインタのマップをサポート

Public Members
- Construction/Destruction:CMapPtrToPtr,˜CMapPtrToPtr
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapPtrToWord (#include 〈afxcoll.h〉) voidポインタによって示される16ビットワードのマップをサポート

Public Members
- Construction/Destruction:CMapPtrToWord,˜CMapPtrToWord
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapStringToOb (#include 〈afxcoll.h〉) ユニークなCStringオブジェクトをCObjectポインタにマップするディクショナリコレクション

Public Members
- Construction/Destruction:CMapStringToOb,˜CMapStringToOb
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapStringToPtr (#include 〈afxcoll.h〉) CStringオブジェクトによって示されるvoidポインタのマップをサポート

Public Members
- Construction/Destruction:CMapStringToPtr,˜CMapStringToPtr
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapStringToString (#include 〈afxcoll.h〉) CStringオブジェクトによって示されるCStringオブジェクトのマップをサポート

Public Members
- Construction/Destruction:CMapStringToString,˜CMapStringToString
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapWordToOb (#include 〈afxcoll.h〉) 16ビットワードによって示されるCObjectポインタのマップをサポート

Public Members
- Construction/Destruction:CMapWordToOb,˜CMapWordToOb
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMapWordToPtr (#include 〈afxcoll.h〉) 16ビットワードによって示されるvoidポインタのマップをサポート

Public Members
- Construction/Destruction:CMapWordToPtr,˜CMapWordToPtr
- Operations:Lookup,SetAt,operator [],RemoveKey,RemoveAll,GetStartPosition,GetNextAssoc
- Status:GetCount,IsEmpty

■CMDIChildWnd Windows MDIのチャイルドウィンドウ機能をサポート

Public Members
- Construction/Destruction:CMDIChildWnd
- Initialization:Create
- Operations:MDIDestroy,MDIActivate,MDIMaximize,MDIRestore,GetParentFrame

Protected Members:Data Members:m_pMDIFrameWnd

■CMDIFrameWnd Windows MDIのウィンドウフレーム機能をサポート

Public Members
- Construction/Destruction:CMDIFrameWnd
- Initialization:Create,CreateClient
- Operations:MDIActivate,MDICascade,MDIGetActive,MDIIconArrange,MDIMaximize,MDINext,MDIRestore,MDISetMenu,MDITile,GetChildFrame
- Data Members:m_hWndMDIClient

■CMemFile (#include 〈afx.h〉) メモリ上のファイルをサポートするCFile-derivedクラス

Public Members:Construction/Destruction:CMemFile,˜CMemFile

■CMemoryException (#include 〈afx.h〉) アウトオブメモリ例外の状態を表示

Public Members:CMemoryException

■CMenu Windowsのメニュー関連の機能の操作をサポート

Public Members
- Construction/Destruction:CMenu,˜CMenu
- Initialization:Attach,Detach,CreateMenu,CreatePopupMenu,LoadMenu,LoadMenuIndirect,DestroyMenu
- Menu Operations:DeleteMenu,TrackPopupMenu
- Menu Item Operations:AppendMenu,CheckMenuItem,EnableMenuItem,GetMenuItemCount,GetMenuItemID,GetMenuState,GetMenuString,GetSubMenu,InsertMenu,ModifyMenu,RemoveMenu,SetMenuItemBitmaps

■CMetaFileDC CDCから派生するメタファイルデバイスコンテキストクラスを定義

Public Members
- Construction/Destruction:CMetaFileDC
- Initialization:Create
- Operations:Close,SelectObject,SelectStockObject

■ModalDialog モデルダイアログボックスをサポート

Public Members
- Construction/Destruction:CModalDialog
- Initialization:CreateIndirect
- Operations:DoModal
- Overridables:OnOK,OnCancel

■CNotSupportedException (#include 〈afx.h〉) 例外処理でサポートされない要求をサポート

Public Members:CNotSupportedException

■CObArray (#include <afxcoll.h>) CObjectポインタ配列をサポート

Public Members
- Construction/Destruction:CObArray, ~CObArray
- Bounds:GetSize, GetUpperBound, SetSize
- Operations:FreeExtra, RemoveAll
- Element Access:GetAt, SetAt:ElementAt
- Growing the Array:SetAtGrow, Add
- Insertion/Removal:InsertAt, RemoveAt
- Operators:operator []

■CObject (#include <afx.h>) 主要な基本クラスを定義

Public Members
- Construction/Destruction: ~CObject, operator new, operator delete
- Diagnostics:AssertValid, Dump
- Serialization:IsSerializable, Serialize
- Miscellaneous:GetRuntimeClass, IsKindOf

Protected Members:CObject

Private Members:CObject, operator =

Macros:RUNTIME_CLASS, DECLARE_DYNAMIC, IMPLEMENT_DYNAMIC, DECLARE_SERIAL Macro, IMPLEMENT_SERIAL Macro

■CObList (#include <afxcoll.h>) CObjectポインタのリストをサポート

Public Members
- Construction/Destruction:CObList, ~CObList
- Head/Tail Access:GetHead, GetTail
- Operations:RemoveHead, RemoveTail, AddHead, AddTail, RemoveAll
- Iteration:GetHeadPosition, GetTailPosition, GetNext, GetPrev
- Retrieval/Modification:GetAt, SetAt, RemoveAt
- Insertion:InsertBefore, InsertAfter
- Searching:Find, FindIndex
- Status:GetCount, IsEmpty

■CPaintDC CDCから派生するデバイスコンテキストクラスを定義

Public Members
- Data Members:m_ps
- Construction/Destruction:CPaintDC, ~CPaintDC

Protected Members:m_hWnd

■CPalette Windows GDIのパレット操作機能をサポート

Public Members
- Construction/Destruction:CPalette
- Initialization:CreatePalette
- Operations:FromHandle, GetPaletteEntries, SetPaletteEntries, AnimatePalette, GetNearestPaletteIndex, ResizePalette

■CPen Windows GDIのペン操作機能をサポート

Public Members
- Construction/Destruction:CPen
- Initialization:CreatePen, CreatePenIndirect
- ~Operations:FromHandle

■CPoint CPoint構造体とPOINT構造体の操作をサポート

Public Members
- Construction/Destruction:CPoint
- Operations:Offset, operator ==, operator !=, operator +=, operator -=
- Operators Returning CPoint Values:operator +, operator -
- Operators Returning CSize Values:operator -

■CPtrArray (#include <afxcoll.h>) voidポインタ配列をサポート

Public Members
- Construction/Destruction:CPtrArray, ~CPtrArray
- Bounds:GetSize, GetUpperBound, SetSize
- Operations:FreeExtra, RemoveAll
- Element Access:GetAt, SetAt, ElementAt
- Growing the Array:SetAtGrow, Add
- Insertion/Removal:InsertAt, RemoveAt
- Operators:operator []

■CPtrList (#include <afxcoll.h>) voidポインタリストをサポート

Public Members
- Construction/Destruction:CPtrList, ~CPtrList
- Head/Tail Access:GetHead, GetTail
- Operations:RemoveHead, RemoveTail, AddHead, AddTail, RemoveAll
- Iteration:GetHeadPosition, GetTailPosition, GetNext, GetPrev
- Retrieval/Modification:GetAt, SetAt, RemoveAt
- Insertion:InsertBefore, InsertAfter
- Searching:Find, FindIndex
- Status:GetCount, IsEmpty

■CRect CRect構造体やWindows RECT構造体の操作をサポート

Public Members
- Construction/Destruction:CRect
- Operations:Width, Height, Size, TopLeft, BottomRight, IsRectEmpty, IsRectNull, PtInRect, SetRect, SetRectEmpty, CopyRect, EqualRect, InflateRect, OffsetRect, IntersectRect, UnionRect
- Operators:operator LPRECT, operator =, operator ==, operator !=, operator +=, operator -=, operator &=, operator |=, operator +, operator -, operator &, operator |

■CResourceException (#include <afxwin.h>) Windowsが要求されたリソースを発見または移動できないときに生じる

Public Members:CResourceException

■CRgn Windows GDIの領域の操作をサポート

Public Members
- Construction/Destruction:CRgn
- Initialization:CreateRectRgn, CreateRectRgnIndirect, CreateEllipticRgn, CreateEllipticRgnIndirect, CreatePolygonRgn, CreatePolyPolygonRgn, CreateRoundRectRgn, CombineRgn, CopyRgn
- Operations:EqualRgn, FromHandle, GetRgnBox, OffsetRgn, PtInRegion, RectInRegion, SetRectRgn

■CScrollBar Windowsのスクロールバーコントロール機能をサポート

Public Members
- Construction/Destruction:CScrollBar
- Initialization:Create
- Operations:GetScrollPos, SetScrollPos, GetScrollRange, SetScrollRange,

■CSize 相対的な座標をインプリメントする

Public Members
- Construction/Destruction:CSize
- Operations:operator==, operator!=, operator+=, operator-=
- Operators Returning CSize Values:operator+, operator-

■CStatic Windowsのスタティックコントロール機能をサポート

Public Members
- Construction/Destruction:CStatic
- Initialization:Create

■CStdioFile (#include <afx.h>) Cのランタイムファイルストリーム機能をサポート

Public Members
- Data Members:m_pStream
- Construction/Destruction:CStdioFile, ~CStdioFile
- Text Read/Write:ReadString, WriteString

■CString (#include <afx.h>) CStringオブジェクトを操作し、色々な関数やオペレーターをサポート

Public Members
- Construction/Destruction:CString, _CString
- The String as an Array:GetLength, IsEmpty, Empty, GetAt, operator [], SetAt, operator const char* ()
- Assignment/Concatenation:operator =, operator +, operator +=
- Comparison:Operators ==, <, etc., Compare, CompareNoCase, Collate
- Extraction:Mid, Left, Right, SpanIncluding, SpanExcluding
- Other Conversions:MakeUpper, MakeLower, MakeReverse
- Searching:Find, ReverseFind, FindOneOf
- Archive/Dump:operator <<, operator >>
- Buffer Access:GetBuffer, GetBufferSetLength, ReleaseBuffer
- Windows-Specific:LoadString, AnsiToOem, OemToAnsi

■CStringArray (#include <afxcoll.h>) CStringオブジェクト配列をサポート

Public Members
- Construction/Destruction:CStringArray, ~CStringArray
- Bounds:GetSize, GetUpperBound, SetSize
- Operations:FreeExtra, RemoveAll
- Element Access:GetAt, SetAt, ElementAt
- Growing the Array:SetAtGrow, Add
- Insertion/Removal:InsertAt, RemoveAt
- Operators:operator []

■CStringList (#include <afxcoll.h>) CStringオブジェクトリストをサポート

Public Members
- Construction/Destruction:CStringList, ~CStringList
- Head/Tail Access:GetHead, GetTail
- Operations:RemoveHead, RemoveTail, AddHead, AddTail, RemoveAll
- Iteration:GetHeadPosition, GetTailPosition, GetNext, GetPrev
- Retrieval/Modification:GetAt, SetAt, RemoveAt
- Insertion:InsertBefore, InsertAfter
- Searching:Find, FindIndex
- Status:GetCount, IsEmpty

■CTime (#include <afx.h>) 日付/時刻の操作をサポート

Public Members
- Construction/Destruction:CTime, GetCurrentTime
- Extraction:GetTime, GetYear, GetMonth, GetDay, GetHour, GetMinute, GetSecond, GetDayOfWeek
- Conversion:GetGmtTm, GetLocalTm, Format, FormatGmt
- Operators:operator =, operators +, -, operators +=, -=, operators ==, < , etc.
- Archive/Dump:operator <<, operator >>

■CTimeSpan (#include <afx.h>) 時間の操作をサポート

Public Members
- Construction/Destruction:CTimeSpan
- Extraction:GetDays, GetHours, GetTotalHours, GetMinutes, GetTotalMinutes, GetSeconds, GetTotalSeconds
- Conversion:Format
- Operators:operator =, operators +, -, operators +=, -=, operators ==, <, etc.
- Archive/Dump:operator <<, operator >>

■CWinApp Windowsアプリケーションのオブジェクトをサポート

Public Members
- Data Members:m_pszAppName, m_hInstance, m_hPrevInstance, m_lpCmdLine, m_nCmdShow, m_pMainWnd
- Construction/Destruction:CWinApp
- Operations:LoadCursor, LoadStandardCursor, LoadOEMCursor, LoadIcon, LoadStandardIcon, LoadOEMIcon
- Overridables:InitApplication, InitInstance, Run, OnIdle, ExitInstance, PreTranslateMessage

Protected Members:m_msgCur

■CWindowDC CDCから派生するウィンドウデバイスコンテキストクラスを定義

Public Members:Construction/Destruction:CWindowDC, ~CWindowDC

Protected Members:m_hWnd

■CWnd Microsoft Foundation Class Libraryのすべてのウィンドウクラスについて基本機能をサポート

Public Members
- Data Members:m_hWnd, wndTop, wndBottom
- Construction/Destruction:CWnd, ~CWnd, DestroyWindow
- Initialization:Create, GetStyle, Attach, Detach, FromHandle, DeleteTempMap, GetSafeHwnd
- Message Functions:SendMessage, PostMessage
- Window Text Functions:SetWindowText, GetWindowText, GetWindowTextLength, SetFont, GetFont
- Menu Functions:GetMenu, SetMenu, DrawMenuBar, GetSystemMenu, HiliteMenuItem
- Child Window Attributes:GetDlgCtrlID
- Window Size and Position:CloseWindow, OpenIcon, IsIconic, IsZoomed, MoveWindow, SetWindowPos, ArrangeIconicWindows, BringWindowToTop, GetWindowRect, GetClientRect
- Coordinate Mapping Functions:ClientToScreen, ScreenToClient
- Update/Painting Functions:BeginPaint, EndPaint, GetDC, GetWindowDC, ReleaseDC, UpdateWindow, SetRedraw, GetUpdateRect, GetUpdateRgn, Invalidate, InvalidateRect, InvalidateRgn, ValidateRect, ValidateRgn, ShowWindow, IsWindowVisible, ShowOwnedPopups
- Timer Functions:SetTimer, KillTimer
- Window State Functions:IsWindowEnabled, EnableWindow, GetActiveWindow, SetActiveWindow, GetCapture, SetCapture, GetFocus, SetFocus, SetSysModalWindow, GetSysModalWindow, GetDesktopWindow
- Dialog-Box Item Functions:CheckDlgButton, CheckRadioButton, GetCheckedRadioButton, DlgDirList, DlgDirListComboBox, DlgDirSelect, DlgDirSelectComboBox, GetDlgItem, GetDlgItemInt, GetDlgItemText, GetNextDlgGroupItem, GetNextDlgTabItem, IsDlgButtonChecked, SendDlgItemMessage, SetDlgItemInt, SetDlgItemText
- Scrolling Functions:GetScrollPos, GetScrollRange, ScrollWindow, SetScrollPos, SetScrollRange, ShowScrollBar
- Window Access Functions:ChildWindowFromPoint, FindWindow, GetNextWindow, GetTopWindow, GetWindow, GetLastActivePopup, IsChild, GetParent, SetParent, WindowFromPoint
- Alert Functions:FlashWindow, MessageBox
- Clipboard Functions:ChangeClipboardChain, SetClipboardViewer, OpenClipboard, GetClipboardOwner, GetClipboardViewer
- Caret Functions:CreateCaret, CreateSolidCaret, CreateGrayCaret, GetCaretPos, SetCaretPos, HideCaret, ShowCaret
- General Message Handlers:OnCommand, OnActivate, OnActivateApp, OnCancelMode, OnChildActivate, OnClose, OnCreate, OnCtlColor, OnDestroy, OnEnable, OnEndSession, OnEnterIdle, OnEraseBkgnd, OnGetMinMaxInfo, OnIconEraseBkgnd, OnKillFocus, OnMenuChar, OnMenuSelect, OnMove, OnPaint, OnPaintIcon, OnParentNotify, OnQueryDragIcon, OnQueryEndSession, OnQueryNewPalette, OnQueryOpen, OnSetFocus, OnShowWindow, OnSize
- Nonclient-Area Message Handlers:OnNcActivate, OnNcCalcSize, OnNcCreate, OnNcDestroy, OnNcHitTest, OnNcLButtonDblClk, OnNcLButtonDown, OnNcLButtonUp, OnNcMButtonDblClk, OnNcMButtonDown, OnNcMButtonUp, OnNcMouseMove, OnNcPaint, OnNcRButtonDblClk, OnNcRButtonDown, OnNcRButtonUp
- System Message Handlers:OnSysChar, OnSysCommand, OnSysDeadChar, OnSysKeyDown, OnSysKeyUp, OnCompacting, OnDevModeChange, OnFontChange, OnPaletteChanged, OnSpoolerStatus, OnSysColorChange, OnTimeChange, OnWinIniChange
- Input Message Handlers:OnChar, OnDeadChar, OnHScroll, OnKeyDown, OnKeyUp, OnLButtonDblClk, OnLButtonDown, OnLButtonUp, OnMButtonDblClk, OnMButtonDown, OnMButtonUp, OnMouseActivate, OnMouseMove, OnRButtonDblClk, OnRButtonDown, OnRButtonUp, OnSetCursor, OnTimer, OnVScroll
- Initialization Message Handlers:OnInitMenu, OnInitMenuPopup
- Clipboard Message Handlers:OnAskCbFormatName, OnChangeCbChain, OnDestroyClipboard, OnDrawClipboard, OnHScrollClipboard, OnPaintClipboard, OnRenderAllFormats, OnRenderFormat, OnSizeClipboard, OnVScrollClipboard
- Control Message Handlers:OnCharToItem, OnCompareItem, OnDeleteItem, OnDrawItem, OnGetDlgCode, OnMeasureItem, OnVKeyToItem
- MDI Message Handlers:OnMDIActivate, GetWindowPlacement, SetWindowPlacement, MapWindowPoints, GetDCEx, LockWindowUpdate, RedrawWindow, ScrollWindowEx, OnPaletteIsChanging, OnWindowPosChanging, OnWindowPosChanged, EnableScrollBar, PostNcDestroy, GetOpenClipboardWindow, OnDropFiles, GetExStyle, SubclassWindow

Protected Members:CreateEx, GetCurrentMessage, GetSuperWndProcAddr, PreTranslateMessage, WindowProc, Default, DefWindowProc

■CWordArray (#include <afxcoll.h>) ワード配列をサポート

Public Members
- Construction/Destruction:CWordArray, ~CWordArray
- Bounds:GetSize, GetUpperBound, SetSize
- Operations:FreeExtra, RemoveAll
- Element Access:GetAt, SetAt, ElementAt
- Growing the Array:SetAtGrow, Add
- Insertion/Removal:InsertAt, RemoveAt
- Operators:operator []

HEWLETT PACKARD



1秒間に、5000回、感じる。

HP独自の
サーマルインクジェット・テクノロジー。
これは、インクの噴射を
1秒間に5000回も行う最新技術と、
厚さ50ミクロンの薄膜フィルムに
レーザで穴をあける
超精密技術から生まれました。
「もっと静かで速くてきれいに」
という願いに、
デスクジェット500Jなら
高感度でお応えします。

独自のサーマルインクジェット方式。

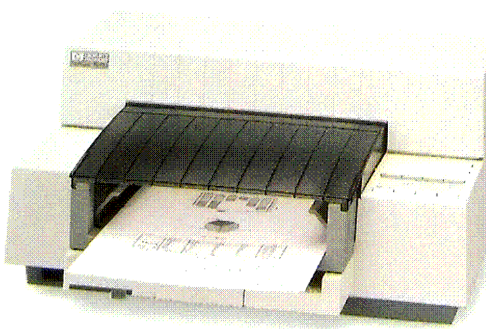
HPが1984年に世界で初めて発売した、熱によりインクを噴射する技術「サーマルインクジェット」方式プリンタ。この実用化を可能にしたのは、インクを噴射するノズルに用いられた高精度の薄膜加工技術です。そしてインクの噴射は、1秒間にじつに5000回。静かで美しいうえに、「とにかく速い」秘密はここにあります。

この価格で、レーザプリンタに迫る。

まず「速さ」。漢字を高品位モードで110字/秒、高速モードにすれば160字/秒。10万円以下では類をみないこの速さが、仕事の能率をグーンとアップさせます。そして44dBの「静かさ」。となりで電話をしていても、夜中でも気兼ねなくプリントできます。さらに300dpiの「美しさ」も魅力。文字もグラフィクスも高品位で印字します。

身近に使いやすい。末永く使いやすい。

「使いやすさ」も、デククジェット500Jの大きな特長です。一度に100枚の用紙をセットできるカットシートフィーダを標準装備。紙の扱いが楽なフロントローディング（前面給紙方式）を採用。そして3年間の長期保証付き。世界中で300万台以上のインクジェットプリンタの販売実績を持つ、HPならではの自信の証しです。



日本語インクジェットプリンタ

HP DeskJet 500J

標準小売価格：98,000円（税別）

製品番号 C2603A

●対応機種
NEC PC-98シリーズ、EPSON・PCシリーズ、東芝J-3100シリーズ、IBM・PC-XT/AT、各社AX、DOS/V仕様パソコン

●別売アクセサリー

インクカートリッジ	51608A	3,200円
大容量インクカートリッジ	51626A	5,200円
丸ゴシックフォントカートリッジ	C3000A	9,800円

お問い合せはフリーダイヤルで
カストマ・コミュニケーションセンタ
0120-081444

横河・ヒューレット・パッカード株式会社
本社 〒168 東京都杉並区高井戸東3-29-21

CPOIP02 AD-2043

〈資料請求番号 P03〉

T1017777080981 〈資料請求番号 P04〉 雑誌コード17777-08